UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

E.A.P DE INGENIERÍA DE SISTEMAS

Implementación de un sistema de integración para las bibliotecas municipales de Lima y Callao utilizando SOA y J2ME Tesina para optar el título profesional de Ingeniero de sistemas

TESIS

para optar el título profesional de Ingeniero de Sistemas

AUTORES

Luis Eduardo Medina Bonilla

Luis Enrique Pinedo Marín

Lima – Perú

2010

FICHA CATALOGRÁFICA

Medina Bonilla, Luis Eduardo
Pinedo Marín, Luis Enrique

“IMPLEMENTACIÓN DE UN SISTEMA DE INTEGRACIÓN PARA LAS BIBLIOTECAS MUNICIPALES DE LIMA Y CALLAO UTILIZANDO SOA Y J2ME".

(Lima) 2010.

viii, 168p., 29.7cm (UNMSM, Ingeniero, Sistemas, 2010)
Tesis, Universidad Nacional Mayor de San Marcos,
Facultad de Ingeniería de Sistemas e Informática.

“A mi abuelita Ramona, a mis padres José y Rosa gracias a ellos por la educación e inconmensurable apoyo que siempre me han brindado en todo momento.”…Luis Medina

“El presente trabajo de investigación está dedicado en primer lugar a Dios, mis padres, mi esposa e hijo y a todas las personas que de alguna manera me apoyaron para lograr la culminación de este trabajo.”...Luis Pinedo

ÍNDICE

RESUMEN

IMPLEMENTACIÓN DE UN SISTEMA DE INTEGRACIÓN PARA LAS BIBLIOTECAS MUNICIPALES DE LIMA Y CALLAO UTILIZANDO SOA Y J2ME
4

MEDINA BONILLA, LUIS EDUARDO

PINEDO MARÍN, LUIS ENRIQUE

Junio - 2010

Asesor : Marcos Sotelo

Título a Obtener : Ingeniero de Sistemas

La presente tesina aborda el tema de Integración de aplicaciones entre los diferentes Sistemas de Bibliotecas Municipales de Lima y Callao, basándose para ello de una arquitectura orientada a servicios (SOA) y como middleware de comunicación un ESB (Enterprise Service Bus), desde el cual también podrá ser accedid4o mediante dispositivos móviles, utilizando la plataforma J2ME (Java 2 Micro Edition).

Palabras Claves:

- SOA
- ESB
- J2ME

ABSTRACT

IMPLEMENTATION OF AN INTEGRATION SYSTEM FOR MUNICIPAL LIBRARIES OF LIMA AND CALLAO USING SOA AND J2ME

MEDINA BONILLA, LUIS EDUARDO

PINEDO MARÍN, LUIS ENRIQUE

Junio - 2010

Adviser : Marcos Sotelo

Degree : Systems Engineer

This thesis is about of the issue of application integration between different systems of public libraries in Lima and Callao, relying on a service-oriented architecture (SOA) and middleware communication as an ESB (Enterprise Service Bus), from which may also be accessed through mobile devices using J2ME (Java 2 Micro Edition).

Key words:

- SOA
- ESB
- J2ME

INTRODUCCIÓN

Hoy en día las organizaciones operan con diversos sistemas informáticos, los cuales tienen que comunicarse entre sí con independencia del tipo de plataforma, para poder intercambiar e integrar la información. En este contexto se hace necesario establecer mecanismos que permitan realizar esta integración para poder ofrecer mejores servicios que puedan brindar una fuente de información diversa y consolidada.

Es importante indicar que esta integración debe estar basada en servicios donde cada aplicación debe exponer su funcionalidad dentro del servicio. Esta integración es posible mediante la implementación de web services.

La presente tesina propone un enfoque de solución basado en una arquitectura orientada a servicios e integración de datos para las bibliotecas municipales de Lima y Callao. Actualmente las bibliotecas municipales de Lima y Callao no cuentan con web services, más bien todas ellas cuentan con aplicaciones que solo proporcionan información de la misma biblioteca, lo cual no resulta cómodo para los usuarios siendo necesario consultar en diferentes bibliotecas.

La presenta tesina plantea una integración basado en servicios web y utilizando un bus de servicios empresarial (ESB), donde cada servicio web será implementado para cada municipalidad que tenga un aplicativo de consulta de material bibliográfico. El bus de servicios será el encargado de dirigir cada petición de consulta hacia el respectivo servicio web.

Las razones que nos llevaron a la elección de la solución son las siguientes:

- Ofrecer un mejor servicio de consulta de material bibliográfico.
- Facilitar a los usuarios de las bibliotecas municipales el acceso a la información de todas las municipalidades sin la necesidad de desplazarse físicamente hasta la misma biblioteca.

La tesina está organizada en capítulos, donde el Capítulo Uno estará dedicado al Planteamiento Metodológico del Problema. El Capítulo Dos presentará el Marco Teórico referente al concepto de las bibliotecas municipales así como de las tecnologías utilizadas para llevar a cabo la integración. El Capítulo Tres estará dedicado al Planteamiento de la Solución, y el Capítulo Cuatro estará dedicado al Desarrollo de la Solución aplicado a un caso. El Capítulo Cinco será para las Conclusiones Y Recomendaciones. En el Capítulo Seis se mostrará el Glosario de Términos y Acrónimos; y en el Capítulo 7 se incluirán las Referencias Bibliográficas.

CAPÍTULO I

1. PLANTEAMIENTO METODOLÓGICO DEL PROBLEMA

1.1 DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA

La palabra biblioteca (del griego βιβλιοθήκη *biblion* = libro y *thekes* = caja), puede traducirse desde un punto de vista estrictamente etimológico como el lugar donde se guardan los libros.

En la actualidad esta concepción ha cambiado desde hace tiempo, para pasar a referirse a un local o centro donde se almacenan libros y publicaciones impresos, u otros tipos de documentos gráficos o audiovisuales, disponibles para préstamos o consultas. La biblioteca tiene como fin resolver los problemas de información de sus usuarios de acuerdo a sus necesidades, entre las cuales se podría mencionar el aprendizaje, la docencia y la investigación.

La Biblioteca Pública es producto de la democracia y una demostración pública de la fe en la educación universal, entendida como proceso que dura toda la vida. Es un centro de información que facilita a sus usuarios toda clase de conocimientos e información. También, podemos decir que es una institución sociocultural que está orientada principalmente a satisfacer las necesidades de información y recreación de la comunidad en la cual está situada, sin distinción de sexo, edad, raza, religión u opciones políticas. La Biblioteca Municipal es un tipo de Biblioteca Pública.

La Biblioteca Pública en el Perú tiene su mejor antecedente en la creación de la Biblioteca Nacional, el 28 de agosto de 1821, impulsada por San Martín, para garantizar la recién ganada libertad, facilitando el acceso a toda la población al conocimiento de todos los tiempos. Más adelante, el Decreto del 8 de febrero de 1822, en su Art. 1º confirmaría la naturaleza de la institución creada, cuando dice: "Se establecerá una Biblioteca Pública con el nombre de Biblioteca Nacional del Perú". .

Se confirma así que la Biblioteca Pública en el Perú nace con la República, como una institución fundamental para la libertad de los hombres y los pueblos o, lo que es lo mismo, como el sustento de una auténtica democracia

El Sistema Nacional de Bibliotecas (SNB) es el ente rector de las Bibliotecas públicas de Perú, Unidades de Información y Centros de Documentación en el ámbito nacional, el SBN es el resultado de la sinergia positiva entre la Biblioteca Nacional del Perú, municipalidades, gobiernos regionales, la sociedad civil; y otras instituciones públicas y privadas, nacionales o internacionales. Tiene entre sus funciones las siguientes:

- Definir, ejecutar y evaluar políticas y programas que coadyuven a la integración de las diversas unidades de información en subsistemas y redes nacionales, regionales y locales así como asegurar la coordinación y cooperación reciproca de sus componentes (...).
- Definir, promover y adoptar una política orientada a la normalización y unificación de los procedimientos técnicos que permitan asegurar una organización y gestión de la información eficiente y eficaz.
- Promover y asegurar acciones y programas para la elaboración y difusión de documentos especializados

Jerárquicamente, el SNB es un organismo técnico de la Biblioteca Nacional del Perú que acredita a los miembros que conforman la red de bibliotecas públicas, asegurando la calidad de sus servicios internos e interbibliotecarios, la accesibilidad de sus catálogos, y la promoción de la lectura en sus comunidades. El Sistema Nacional de Bibliotecas sustenta su labor en su propia normatividad en coordinación con la Biblioteca Nacional del Perú. Además, participa en la conformación de redes de bibliotecas escolares, universitarias y especializadas mediante procesos de consultoría en materia normativa o formativa.

A continuación mostramos la estructura orgánica del SNB

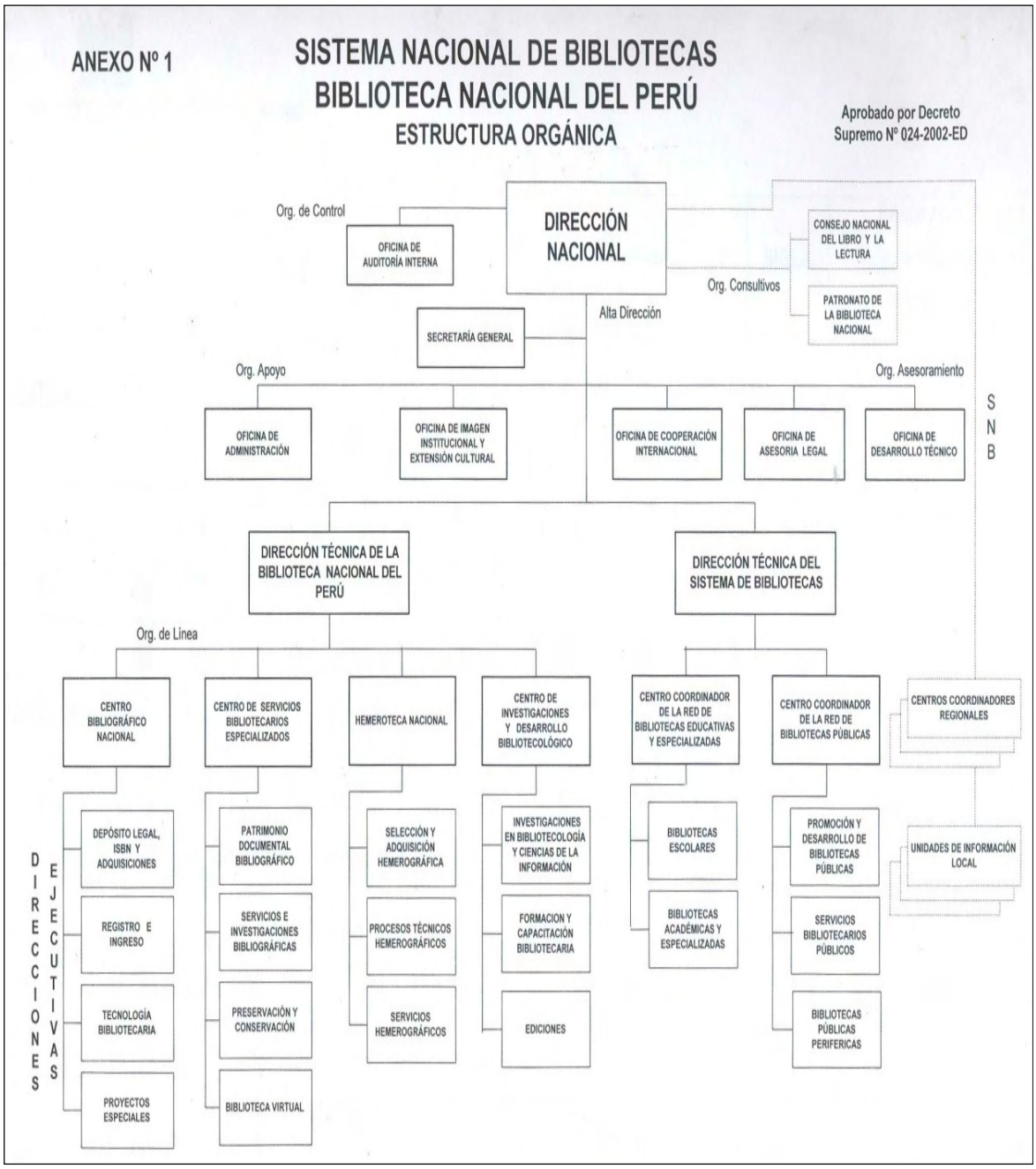


Figura 1.1 Estructura Orgánica Sistema Nacional de Bibliotecas

En nuestro país existen algunas municipalidades que brindan bibliotecas para el acceso de los pobladores de un determinado ámbito geográfico, a este tipo de biblioteca se denomina biblioteca municipal. Actualmente estas bibliotecas municipales cuentan con sistemas computarizados de búsqueda de información del material bibliográfico, de los cuales algunos son sistemas Cliente/Servidor y otros son sistemas Web. Los bibliotecas que poseen un sistema Web, algunas usan como Base de Datos el WINISIS (Software desarrollado por la UNESCO) y otras cuentan con un DBMS específico.

La principal limitación de estos sistemas es que la información consultada corresponde únicamente a la biblioteca municipal. Por ejemplo si un usuario no encontrará un libro en una determinada biblioteca, tendría que ir a consultar a otra, hasta encontrar el libro. Esto trae como consecuencia inconvenientes y molestias al usuario al momento de ubicar un material bibliográfico.

Como se puede notar, se hace necesario resolver esta limitación, motivo por el cual se propondrá una solución a esta problemática haciendo uso de una óptima integración tecnológica entre todas las bibliotecas municipales, independientemente del lenguaje en que fue desarrollado el sistema de cada biblioteca municipal.

Actualmente se viene desarrollando un proyecto llamado “Catálogo Unido Automatizado de las Bibliotecas Públicas del Sistema Nacional de Bibliotecas del Perú, es la integración del conjunto de catálogos bibliográficos automatizados pertenecientes a Bibliotecas Públicas del país, que en una primera etapa consta de los registros de las bibliotecas de Lima, Callao, Capitales de Departamento y de las seis Bibliotecas Periféricas de la Biblioteca Nacional del Perú. Esta integración consiste en registrar las colecciones bibliográficas de las Unidades de información de las Bibliotecas Públicas del país, centralizando en un solo lugar toda la información de todas las bibliotecas públicas del Perú, una desventaja de este proyecto es el que si la municipalidad tuviera ya un sistema desarrollado con Base de datos propia, tendría que migrar toda su información a la Base de Datos Central, y los nuevos registros bibliográficos tendrían que ser enviados hasta la sede central para su actualización.

Actualmente las bibliotecas municipales de Lima y Callao que cuentan con un sistema de biblioteca son las siguientes:

1. Biblioteca Municipal de Barranco
2. Biblioteca Municipal de Bellavista
3. Biblioteca Municipal de Callao
4. Biblioteca Municipal de Huaral
5. Biblioteca Municipal de Independencia
6. Biblioteca Municipal de Jesús María
7. Biblioteca Municipal de La Perla
8. Biblioteca Municipal de La Punta
9. Biblioteca Municipal de Lima
10. Biblioteca Municipal de Lince
11. Biblioteca Municipal de Magdalena del Mar
12. Biblioteca Municipal de Miraflores
13. Biblioteca Municipal de Rímac
14. Biblioteca Municipal de San Borja
15. Biblioteca Municipal de San Isidro
16. Biblioteca Municipal de San Luis
17. Biblioteca Municipal de San Miguel
18. Biblioteca Municipal de Santiago de Surco
19. Biblioteca Municipal de Ventanilla

1.2. DEFINICIÓN DEL PROBLEMA

El problema principal y problemas secundarios son:

1.2.1. PROBLEMA PRINCIPAL

El problema es la falta de integración entre las bibliotecas municipales lo que ocasiona que el usuario tenga que ir de una biblioteca a otra para ubicar el material bibliográfico que desea.

1.2.2. PROBLEMAS SECUNDARIOS

- Las bibliotecas municipales no presentan una implementación de servicios web que permita exponer la información bibliográfica que poseen.

- Las bibliotecas municipales en algunos casos cuenta con aplicativos que funcionan dentro del local lo que limita el acceso a la información.

- Las bibliotecas tienen un determinado horario de atención y poco personal lo que limita la disponibilidad de la información.

1.3. OBJETIVOS

El objetivo general y objetivos específicos son:

1.3.1. OBJETIVO GENERAL

Implementar un sistema de integración para las bibliotecas municipales de Lima y Callao utilizando una arquitectura orientada a servicios donde cada biblioteca haga disponible su información a través de un servicio web y donde se alcance la integración mediante una plataforma ESB.

1.3.2. OBJETIVOS ESPECÍFICOS

- Proporcionar a las bibliotecas municipales una extensa y poderosa herramienta de consulta a través de internet donde el usuario pueda navegar por cualquier biblioteca en búsqueda de la información que desee.

- Brindar al usuario la posibilidad de acceder a este sistema de consulta a través de cualquier dispositivo móvil haciendo uso de la plataforma J2ME.

- Implementar esta solución utilizando software libre, lo cual permitirá su aceptación en las municipalidades debido al ahorro considerado de costos.

1.4. JUSTIFICACIÓN E IMPORTANCIA

Este proyecto de investigación tiene una muy fuerte justificación práctica ya que en nuestro país existen un gran número de bibliotecas municipales donde la información es centralizada respecto al ámbito de la biblioteca, y donde no existe la integración, generando molestias e inconvenientes a las personas que desean realizar algún estudio o investigación.

La importancia de este proyecto radica en que si aplicamos la integración de las bibliotecas municipales se logrará brindar un mejor servicio al usuario que desea ubicar cierto material bibliográfico, apoyando en cierto grado en la facilitación de la

investigación y estudio en el país. Cabe mencionar también que este modelo de integración se podría aplicar en otros rubros, como por ejemplo integración de entidades del gobierno, universidades, empresas privadas y cualquier necesidad de integración.

1.5. LIMITACIONES Y ALCANCES

Las limitaciones y alcances de la solución propuesta son:

1.5.1. LIMITACIONES

- El servicio web a implementar en cada biblioteca municipal se restringe solamente a la consulta de material bibliográfico.

- Se implementará el servicio web en cada biblioteca municipal que tenga ya disponible un Sistema de Bibliotecas.

- Para el acceso al sistema de biblioteca municipal vía dispositivo móvil se consideran sólo aquellos dispositivos que puedan ejecutar aplicaciones Java y que tengan conexión disponible a internet.

- El componente central de la integración de los sistemas (ESB) radicará en un único ambiente que podría ser la Biblioteca Nacional del Perú.

1.5.2. ALCANCES

- El alcance de la investigación abarca sólo las Bibliotecas Públicas Municipales, por los siguientes motivos:

    1 Las municipalidades están obligadas a sostener bibliotecas en sus jurisdicciones (Ley No 27972; Ley Orgánica de Municipalidades).
    2 Las municipalidades cuentan con recursos presupuestados permanentes y capacidad de intervención en sus comunas.
    3 Las municipalidades brindan servicios al público en general y desarrollan actividades culturales.
    4 Las municipalidades cuentan con mayor soporte logístico e informático, así como canales de coordinación y comunicación.

- La solución de integración será propuesta para las bibliotecas municipales de Lima y Callao.

- La solución es flexible a expandir su rango de cobertura pudiendo considerar bibliotecas universitarias, de colegios, privadas entre otros.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 BIBLIOTECAS PÚBLICAS EN EL PERÚ

La Biblioteca Pública es producto de la democracia y una demostración pública de la fe en la educación universal, entendida como proceso que dura toda la vida. Es un centro de información que facilita a sus usuarios toda clase de conocimientos e información. También, podemos decir que es una institución sociocultural que está orientada principalmente a satisfacer las necesidades de información y recreación de la comunidad en la cual está situada, sin distinción de sexo, edad, raza, religión u opciones políticas

La Organización de las Naciones Unidas para la Educación, la Ciencia y la cultura, UNESCO, a través de sus tres sucesivos Manifiestos sobre bibliotecas públicas, ha contribuido enormemente al desarrollo de esta biblioteca, perfeccionando y actualizando periódicamente las bases teóricas y proponiendo una serie de orientaciones para su establecimiento y expansión como un servicio básico, sostenido por el Estado, con participación de la comunidad. Debe anotarse que, de los tres Manifiestos publicados por la UNESCO en los últimos cincuenta años, por lo menos los dos últimos, han sido elaborados con apoyo de la Federación Internacional de Asociaciones de Bibliotecarios y de Bibliotecas, FLA, entidad a la que igualmente se debe extender el reconocimiento por su permanente preocupación por el desarrollo de la biblioteca pública.

El primer Manifiesto de la UNESCO de 1949, redactado por el prestigioso escritor francés André Maurois, planteaba que la ***"la Biblioteca Pública es un producto de la moderna democracia y una demostración pública de la fe en la educación universal como un proceso que dura toda la vida",*** concibiéndola como un servicio gratuito, una fuerza vital de la colectividad, destinado básicamente al adulto como una universidad del pueblo, pero también que sirva de apoyo a la escuela y a los niños, para formar en ellos el gusto por la lectura, en un enfoque de educación permanente.***(Consultar Referencia URL[1])***

Asimismo, como no podía ser de otra forma, en cuanto a su administración, propone que la Biblioteca Pública debe ser sostenida por el Estado, con fondos públicos.

En 1972, con motivo del Año Internacional del Libro, la UNESCO aprueba un segundo Manifiesto, proponiendo una Biblioteca Pública como una ***"fuerza viva al servicio de la educación, la cultura y la información como instrumento indispensable para el fomento de la paz y de comprensión entre las personas y entre las naciones".*** De nuevo desataca el carácter público del servicio de la biblioteca y la ubica dentro del proceso de educación permanente del individuo, facilitando ***el "libre acceso a la suma de conocimientos y de las ideas del hombre y a las creaciones de su imaginación",*** actuando en la práctica como un centro cultural de la comunidad. Reconoce del mismo modo la importancia del libro y de los otros materiales impresos, pero también, por primera vez, de las ***"nuevas formas de soporte para la información que ocuparán un lugar cada vez más importante entre los fondos de las bibliotecas públicas"***. Propone la diversificación de los servicios de las bibliotecas, en función con el tipo de usuario: niños. Estudiantes, minusválidos, minorías lingüísticas y en general la comunidad. ***(Consultar Referencia URL[2])***

El último Manifiesto de la UNESCO, en noviembre de 1994, propone la Biblioteca Pública como: ***"La biblioteca pública es un centro de información que facilita a los usuarios todo tipo de datos y conocimientos. La biblioteca pública presta sus servicios sobre la base de igualdad de acceso de todas las personas, independientemente de su edad, raza, sexo, religión, nacionalidad, idioma o condición social. Debe contar además con servicios específicos para quienes por una u otra razón no puedan valerse de los servicios y materiales ordinarios, por ejemplo, minorías lingüísticas, deficientes físicos y mentales, enfermos o reclusos".*** Así pues, la UNESCO alienta a las autoridades nacionales y locales a que apoyen las bibliotecas públicas y participen activamente en su desarrollo ***(Consultar Referencia URL [3])***

La biblioteca pública cumple las siguientes funciones:

- Mantener la democracia, al proporcionar el acceso a todo material de información.
- Apoyar el sistema educativo y de aprendizaje.

- Ofrecer a su comunidad la oportunidad de utilizar dichos servicios con nuevas tecnologías de amplia aplicación.
- Actuar como Institución cultural.
- Mantener y desarrollar la calidad de vida de sus comunidades.

La biblioteca pública de hoy no sólo deberá prestar los servicios y materiales convencionales sino, además deberá hacer uso y poner al servicio de su comunidad computadoras, con las que se podrá acceder a ficheros o base de datos de cualquier parte del mundo. Esta Biblioteca tendrá como herramienta principal equipos tecnológicos, cambiando, de esta manera la actitud de la población y de aquellos a quienes preste servicio

El parlamento Europeo considera que "…tener y saber utilizar la información constituye un factor de integración económica, social y cultural y que, por lo tanto, es conveniente organizar y garantizar el libre acceso de los ciudadanos" a la información". Por lo tanto, la biblioteca pública del futuro tendrá que tomar en cuenta lo siguiente:

- Tener catálogos colectivos
- Integrarse en red
- Cooperar con otras instituciones
- Ofrecer desde servicios especiales de información empresarial hasta servicios a minorías étnicas o personas discapacitadas, sin distinción alguna.

La Biblioteca Pública en el Perú; el libertador don José de San Martín, fundó la Biblioteca Pública de Lima o Biblioteca Nacional, para uso exclusivo de todas las personas que gusten concurrir en ella, cuyo primer director fue el sacerdote Mariano Teodoro José de Arce.

Carmen Checa de Silva, Directora de la Oficina de Bibliotecas Públicas de la Biblioteca Nacional hasta 1986 dividió la Historia de la Biblioteca Pública en el Perú en cinco periodos: ***(Consultar Referencia Libros [1])***

1er Periodo: La Biblioteca Nacional y la Biblioteca Obrera (1821-1921)

Una Vez proclamada la independencia del Perú, una de las mayores preocupaciones del general don José de San Martín fue la educación y la cultura por lo que funda la Biblioteca Pública de Lima el 28 de agosto de 1821, donando su biblioteca personal.

2do Periodo: La Biblioteca Pública Municipal (1922-1946)

La Biblioteca Pública Municipal es aquella institución que depende o se encuentra regentada por el Consejo Municipal. La biblioteca pública en el Perú marca sus inicios al promulgarse la Ley Nro 4506 de 1922, que obliga a las municipalidades provinciales a brindar este servicio.

3er Periodo: El fondo San Martín (1947-1970)

En 1947 se promulgó la ley Nro 10847, que creó el impuesto a la compra de joyas, lo que originó el fondo económico que sirvió para financiar la construcción del nuevo local de la biblioteca. El fondo se denominó San Martín y fue administrado por el Ministerio de Educación. Una vez terminada la obra, el fondo sirvió para subvencionar la construcción de bibliotecas populares municipales en las provincias y distritos del país.

4to Periodo: Participación Vecinal – Biblioteca Vulga (1971-1979)

Se originó con una profunda transformación de la sociedad peruana. Se crearon bibliotecas en zonas urbano-marginales y rurales, que tomaron diferentes denominaciones, pero que tenían un origen común: la participación de los miembros del lugar para crear un centro cultural que satisficiera sus necesidades educativas, recreacionales y culturales. Por ello se clasificó en tres clases:
Popular, Rural y Comunal

5to Periodo: La Automatización de la Biblioteca (1980-2000)

Uno de los mayores logros del hombre en la actual era moderna es sin lugar a dudas el computador. Estos diminutos cerebros electrónicos han sido capaces de disminuir el tiempo en las actividades productivas, administrativas y la toma de decisiones, permitiendo que el desarrollo de la humanidad se acelere.

La sociedad actual es una sociedad informatizada y la clave está en los computadores; es difícil pensar una profesión en la que no haya influido. En la década de los treinta cuando la Biblioteca de la Universidad de Texas llevó a cabo el control de préstamo de libros utilizando para ello el procesamiento de dicha

información con tarjetas perforadas. En 1967 la Biblioteca del Congreso de los Estados Unidos de Norteamérica. Diseñó el programa MARC (Machine Readable Cataloging) empleando cintas magnéticas para su distribución de su base de datos. Este mismo año la Universidad de Ohio (EE.UU.), estableció la catalogación en línea, fichas catalográficas, listados en microformas para servir a las bibliotecas de su estado.

En la década del setenta, la UNESCO creó el programa Integrated Set of Information Systems (ISIS) para ser utilizado por los computadores de la primera generación. Con la invención del Chip, nacen los microprocesadores y las microcomputadoras por lo que la UNESCO decide adaptar el ISIS en éstas denominándolas MICROISIS. El computador ha agilizado y mejorado los procesos en las bibliotecas, permitiendo la recuperación del documento primario y, por ende, la producción de documentos secundarios.

En el año de 1985 la Dirección de Bibliotecas Públicas de la Biblioteca Nacional del Perú con el apoyo del Centro Panamericano de Ingeniería Sanitaria (CEPIS) da inicio al trabajo automatizado utilizando un computador con sistema Wang 2200 y software DBase II para elaborar un Tesauro y directorio de bibliotecas públicas.
A mediados de los ochenta, la BNP recibe una donación de la IBM que consistió en un computador IBM 36. Mainframe donde se empezaron a llevar a cabo los primeros intentos para procesar la información de manera automatizada empleando para ello el software ISIS y el formato MARC. En 1993, la BNP puso a disposición de los usuarios sus catálogos en línea utilizando para ello una Pentium Pro II, el software MicroIsis, formato MARC y una red tipo Novell. En 1993, la Biblioteca Pública Municipal de Miraflores, bajo la dirección de la Bib. Rosario Prado, dio inicio al proyecto de automatización de su colección. En 1994, la municipalidad remodela su local e inaugura el Centro Cultural Ricardo Palma, brindando los servicios de biblioteca y cabinas públicas de internet a cargo de los profesionales del CEPIS.

La Biblioteca Municipal es un tipo de Biblioteca pública que se rige por legislación específica y cuya administración y presupuesto lo asumen los gobiernos locales, como sucede en el Perú, cabe mencionar que la primera Biblioteca Pública del Perú fue la “José de San Martín” creada en Ica en 1853 para servir a la población que día a día crecía en número y en necesidades básicas como son los libros ya que *“Un pueblo sin cultura, es un pueblo que duerme”*

A continuación mostramos un gráfico comparativo sobre el nivel de cobertura de cuántas bibliotecas municipales tiene el Perú por departamento

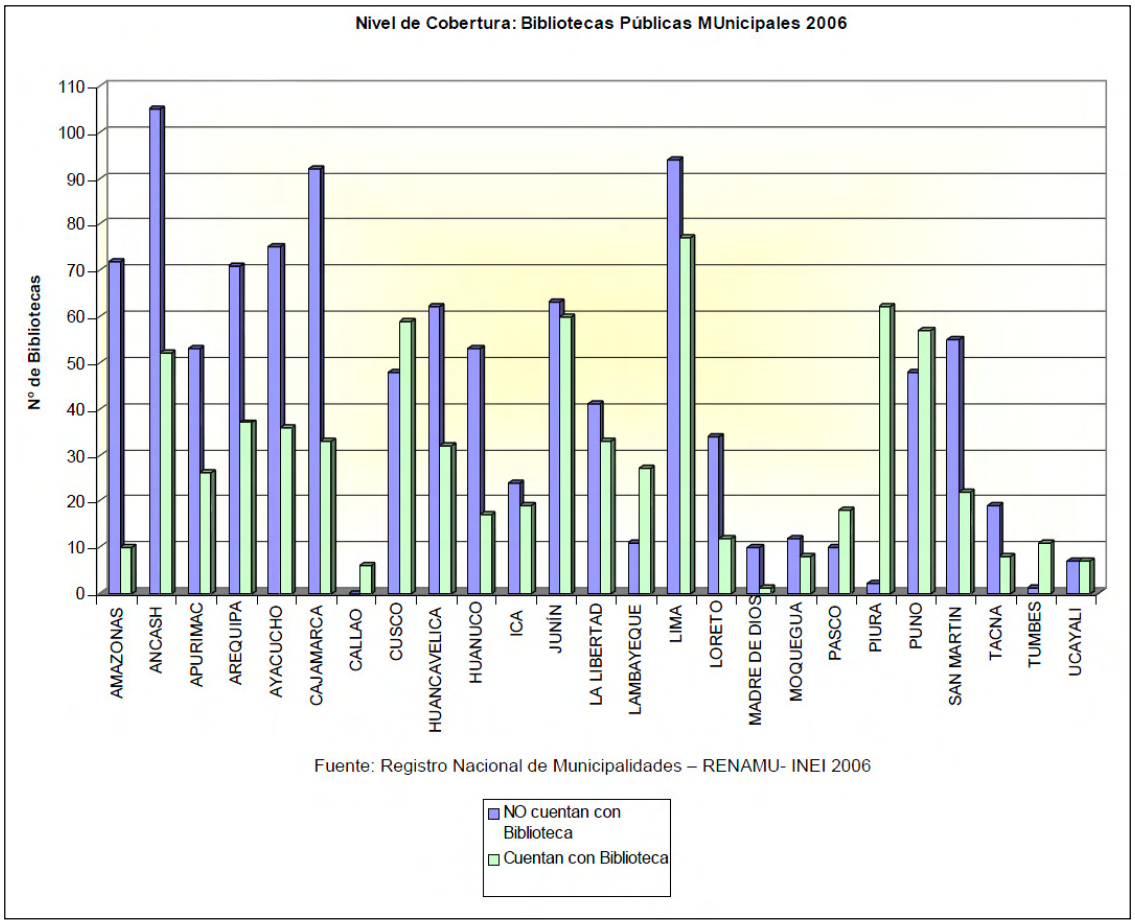


Figura 2.1 Departamentos que cuentan con Bibliotecas Municipales

Colecciones (Libros y folletos) en las Bibliotecas Municipales Años 2004-2006 Según Departamentos

Región	COLECCIONES EN LA BIBLIOTECA MUNICIPAL							
	RENAMU 2004		RENAMU 2005		RENAMU 2006		Promedio	%
	Libros	%	Libros	%	Libros	%		
AMAZONAS	5760	0.5%	3397	0.3%	10528	0.7%	6562	0.5%
ANCASH	57931	4.6%	65983	5.0%	95784	6.5%	73233	5.4%
APURIMAC	17652	1.4%	24387	1.9%	40001	2.7%	27347	2.0%
AREQUIPA	76060	6.0%	72827	5.5%	97125	6.6%	82004	6.0%
AYACUCHO	22809	1.8%	31589	2.4%	36896	2.5%	30431	2.2%
CAJAMARCA	40405	3.2%	48280	3.7%	60876	4.1%	49854	3.7%
CALLAO	56658	4.5%	62935	4.8%	68864	4.7%	62819	4.6%
CUSCO	99120	7.8%	101406	7.7%	110987	7.5%	103838	7.7%
HUANCAVELICA	25940	2.0%	23639	1.8%	40949	2.8%	30176	2.2%
HUÁNUCO	11557	0.9%	6025	0.5%	10937	0.7%	9506	0.7%
ICA	33469	2.6%	31805	2.4%	35656	2.4%	33643	2.5%
JUNÍN	72193	5.7%	89628	6.8%	89774	6.1%	83865	6.2%
LA LIBERTAD	63320	5.0%	42684	3.2%	50793	3.4%	52266	3.9%
LAMBAYEQUE	43990	3.5%	49608	3.8%	49706	3.4%	47768	3.5%
LIMA	326744	25.7%	358696	27.3%	345377	23.3%	343606	25.3%
LORETO	38215	3.0%	30375	2.3%	44478	3.0%	37689	2.8%
MADRE DE DIOS	60	0.0%	2000	0.2%	2020	0.1%	1360	0.1%
MOQUEGUA	12146	1.0%	5902	0.4%	9094	0.6%	9047	0.7%
PASCO	21030	1.7%	19618	1.5%	23249	1.6%	21299	1.6%
PIURA	110301	8.7%	124883	9.5%	134742	9.1%	123309	9.1%
PUNO	53213	4.2%	56503	4.3%	62354	4.2%	57357	4.2%
SAN MARTÍN	61586	4.8%	39634	3.0%	26292	1.8%	42504	3.1%
TACNA	4620	0.4%	2794	0.2%	5582	0.4%	4332	0.3%
TUMBES	5223	0.4%	9736	0.7%	14665	1.0%	9875	0.7%
UCAYALI	11484	0.9%	11459	0.9%	13148	0.9%	12030	0.9%
Total	**1271486**	**100.0%**	**1315793**	**100.0%**	**1479877**	**100.0%**	**1355719**	**100.0%**

Figura 2.2: Cantidad de Libros por Departamento

Fuente: Registro Nacional de Municipalidades (RENAMU) INEI 2004-2006

2.2 APLICACIONES ISIS

2.2.1 WINISIS (CDS/ISIS)

CDS/ISIS para Windows, como su nombre lo indica es un sistema basado en Windows. Los programas para Windows poseen varias características distintivas dadas por el propio sistema operativo Windows. Se define a Microsoft Windows como un entorno gráfico para el usuario, característica que le otorga un mayor control sobre la forma de trabajar así como también permite al usuario una mejor utilización del poder de su PC.

Desde la aparición temprana de la versión DOS, este programa se orientó al manejo de información bibliográfica, esto es, información acerca de documentos tales como libros, artículos de periódicos, actas de conferencias. Normalmente cada registro en una base de datos contiene información acerca de un documento. Muchas de las características de CDS/ISIS son diferentes de las de aquellos sistemas de manejo de bases de datos que han sido diseñados para fines generales.

Técnicamente hablando, la principal característica de CDS/ISIS que lo diferencia de otros sistemas de manejo de base de datos es la utilización de campos de texto de longitud variable. La información (datos) en un sistema de manejo de base de datos se ingresa en campos. En muchos paquetes de manejo de base de datos tales como dBase o Microsoft Access, los campos son de longitud fija. Es más sencillo diseñar un sistema donde los campos tienen longitud fija y, para muchas aplicaciones, eso no constituye un problema. En un sistema de manejo de personal, se pueden abreviar los ítems de cada individuo para adecuarlo a la longitud del campo disponible o aplicar la utilización de códigos. En sistemas de manejo financiero, se pueden utilizar códigos reemplazando productos, y su número o valor puede almacenarse en un número limitado de dígitos, y de esta forma los datos de longitud variable no son un requisito para dichos sistemas.

Los datos de tipo bibliográfico tienden a ser tratados en forma diferente de otros tipos de datos, valiéndose de menor cantidad de abreviaturas. Más aún, los títulos de libros y otros trabajos contenidos en un registro bibliográfico pueden tener cualquier longitud, que puede variar desde una hasta muchas palabras. Debido a ello, debía hallarse un método que permitiera manejar campos de longitud variable,

característica que muchos paquetes de bases de datos no permiten. Esto es posible gracias a la ayuda de un directorio, el cual se halla también en el formato ISO 2709 (es un estándar ISO para la descripción bibliográfica), en el formato MARC y en los otros formatos que en él se basan .Al comienzo de cada registro existe una lista de campos y punteros que apuntan directamente a la posición del dato que pertenece a cada campo.

Otra de las características de las bases de datos bibliográficas es la necesidad de manejar campos repetibles. Un libro puede tener varios autores. Cada autor debe tener la misma jerarquía. Muchas bases de datos bibliográficas desarrolladas sobre sistemas de manejo de bases de datos tradicionales definen un campo para 'autores'; vale decir que todos los autores de un libro se ingresan en el mismo campo, pero sólo el primero es recuperable. En CDS/ISIS, cada atributo que puede tener más de un valor se ingresa en su propio campo. En otras palabras, cada campo puede repetirse – hasta un límite de 999 veces.

Los datos de tipo bibliográficos pueden hacer uso extensivo de subcampos. Esta facilidad está disponible en el formato ISO 2709 y también la ha implementado CDS/ISIS. Es una característica sumamente útil para campos que deben dividirse en diversas partes para ser tratados de diferente forma. El nombre de un autor puede aparecer en un índice en la forma Smith, John, pero en otras puede ser necesario imprimirlo como John Smith, para producir por ejemplo, salidas en diferentes estilos de referencia.

La existencia de subcampos permite que se pueda manipular en forma separada las diferentes partes del nombre. Se identifican por medio de una letra y, al ingresar el dato en CDS/ISIS, usted antepone el signo ^ (circunflejo) a la letra que identifica el subcampo. Por ello debe ingresar, por ej. el nombre Simon Maxwell como ^aMaxwell^bSimon. El sistema puede tratar a ambas partes individualmente y puede presentarlas en cualquier orden, colocar puntuación entre las partes, escribir en mayúsculas una parte y no otra y así sucesivamente. Los formatos de intercambio mencionados anteriormente prevén reglas para la formulación de campos y subcampos.

Quienes critican la utilización de subcampos objetan la dificultad en el ingreso de los datos. Sin embargo, CDS/ISIS para Windows tiene un sistema de ayuda que permite visualizar mensajes para cada campo particular dentro de la Hoja de

Ingreso de Datos que consiste en dos líneas de mensaje de ayuda al pie de la ventana de ingreso. La existencia de subcampos es de gran utilidad para la organización y manejo de datos. Permiten que elementos de dato repetidos se puedan asociar correctamente con otros elementos de dato. Si los datos de nombre y apellido se ingresaron en campos separados, se requeriría de un mecanismo más complicado para asociar cada nombre con su apellido correspondiente.
CDS/ISIS utiliza la estructura de archivos invertidos para agilizar la búsqueda en una base de datos. Un archivo invertido es un nombre alternativo que se da a un archivo de índice. Esta expresión hace referencia al hecho de que los registros se invierten para que los elementos contenidos en los campos sean accesibles como palabras índice en un archivo.

Es posible indizar cada campo de diferentes formas aplicando distintas técnicas de indización: por el contenido del campo completo, por el contenido de cada subcampo en forma individual, o por cada palabra. Existen además 2 técnicas de indización que indizan el texto encerrado entre signos <....> o /.../. Esta flexibilidad en la indización es poco frecuente en otros sistemas de manejo de base de datos.
Además, es posible efectuar búsquedas de cadenas de texto almacenadas en cualquier campo o buscar en campos numéricos valores superiores, iguales o inferiores a un valor dado. Usted también puede buscar aquellos registros con o sin contenido en algún campo en particular.

Una característica adicional de este paquete es la flexibilidad en el manejo de su visualización en pantalla y salidas de impresión. Esto es posible gracias a un sofisticado manejo de tipo algebraico de su lenguaje de formateo. El mismo ha sido criticado debido a ser complejo y poco amigable, El lenguaje de formateo sirve a un gran número de propósitos además de su función de proveer instrucciones para la visualización en pantalla y salidas impresas.

- Se utiliza para especificar las reglas para la extracción de datos de los registros de la base de datos para generar el índice.
- Se utiliza para la extracción de datos con el objeto de exportarlos hacia otras bases de datos o para convertir los registros a formato MARC cuando los mismos no han sido ingresados siguiendo las reglas del formato MARC.
- Se utiliza como base del lenguaje de búsqueda con el objeto de proveer al paquete de una característica sumamente poderosa de búsqueda en texto libre, incluyendo la búsqueda de valores mencionados anteriormente.

- Se utiliza en la declaración de cláusulas para un archivo de validación de entrada de datos. Nótese que esta característica fue introducida para CDS/ISIS para Windows y no está disponible en la versión de DOS.

El lenguaje de formateo provee a los usuarios de CDS/ISIS de un nivel más alto de control sobre sus datos comparado con el que probablemente encuentre en cualquier otro paquete de automatización de bibliotecas de tipo comercial.

Otra característica importante, considerando que el programa es un producto de UNESCO, es la naturaleza multilingüe del paquete. Tanto los textos de los menús como las hojas de trabajo podrán ser cambiados fácilmente por un usuario avanzado. Los archivos de mensajes se almacenan como bases de datos y pueden, por lo tanto, editarse como cualquier otra base de datos. Pueden modificarse para adecuarla a la utilización de diferente terminología o adaptarse a diferentes idiomas.

No es necesario que los usuarios conozcan las características técnicas del paquete para saber operarlo (y este Manual no apunta en primera instancia al usuario con alto nivel de conocimientos técnicos). Sin embargo, esto es de gran ayuda, ya que una amplia comprensión de la terminología específica puede resultar beneficiosa particularmente cuando las cosas funcionan mal. (Afortunadamente esto no ocurre a menudo con usuarios que utilizan el CDS/ISIS en una forma directa).

CDS/ISIS tiene protección de Copyright y no es, en forma alguna, un programa shareware ni de dominio público.
Puede utilizarse en forma legal mediante el otorgamiento de una licencia la cual no llega a ser tan restrictiva como de hecho lo son muchos productos de software producidos comercialmente, permitiéndose la copia múltiple dentro de una misma institución que sea también titular de licencia. No obstante, no se debería permitir la copia a personas o instituciones que no posean dicha licencia. Una vez obtenida la misma, es válida para todas las versiones, y de hecho las nuevas versiones podrán obtenerse legalmente a través de cualquier medio.

UNESCO no distribuye el software desde su sitio Web.

Aunque UNESCO es la responsable total del desarrollo del paquete CDS/ISIS, la misma organización ha designado distribuidores en diferentes partes del mundo

para colaborar en la tarea de distribuirlo a quienes deseen utilizarlo. En muchos casos, estos distribuidores son los propios puntos focales del Programa Intergubernamental de la UNESCO para el Programa General de Información. Si usted solicita a UNESCO una copia del programa de CDS/ISIS y en el país en el que se encuentra existe un distribuidor nacional, su pedido será canalizado a través de dicho distribuidor. En Perú, el distribuidor nacional es el CONCYTEC.

2.2.2 WWWISIS

Es un programa desarrollado y distribuido por BIREME/PAHO/WHO, especialmente diseñado para operar bases de datos ISIS en servidores WWW (World Wide Web) en un entorno cliente/servidor.

El programa fue confeccionado para realizar búsqueda y entrada de datos, esto significa que permite hallar los datos en las bases en forma rápida, segura y sobre todo eficiente, pudiendo darle las mismas características de búsqueda que el MicroISIS. También permite ingresar los datos en la base a través de Internet en forma remota.

Para poder utilizar el programa se debe contar con ciertos conocimientos previos de HTML (HyperText Markup Language), CGI (Common Gateway Interfase), ISIS, motor de búsqueda WWWISIS, y de cómo interactúan las aplicaciones cliente/servidor de Internet. Este debe ser el punto de partida que debe tener en cuenta cualquier futuro desarrollador. Este programa funciona bajo los entornos UNIX y DOS.

Para poder interactuar con el motor de búsqueda WWWISIS se debe utilizar interfases CGI que servirán para que el usuario ingrese los datos a buscar.

Junto a los datos del usuario se incorporan los parámetros confeccionados por el programador que permiten que dichos datos sean comprendidos por el WWWISIS. Estos constituirán un script CGI que puede contener como parte de su estructura una o más llamadas al WWWISIS, especificando cual va a ser la operación a realizar en la base y con qué formato se visualizará. Esta operación es manejada a través de parámetros que son especificados en la línea de comando por medio de un archivo.

En respuesta a las expresiones agrupadas a través de formatos HTML, pasados vía CGI al WWWISIS, se realizará la búsqueda en la base de datos y luego el resultado formateado será enviado al cliente.

WWWISIS trabaja estrechamente ligado a aplicaciones CGI, que es una parte integrante del servidor WWW, siendo un mecanismo para llamar otros programas. Requiere desde un browser hasta un servidor de Web para que pueda enviar un

programa o script, el cual puede traer datos desde una base de datos y envía la salida al servidor Web, que a su vez lo transfiere al browser, en formato HTML.

Los programas o scripts CGI residen en uno o más directorios reconocidos por el servidor Web como caminos del CGI. El camino del CGI está configurado por el administrador del servidor WWW y usualmente el nombre del directorio es /cgi-bin/ que esta en el directorio del servidor. En consecuencia todos los programas o scripts pueden ser llamados vía cgi a ese camino. El modo más común de reunir datos en un browser es a través del form. El form es un elemento del lenguaje HTML que permite la entrada de datos, tal como campo texto, list box, check box, radio box , etc. Una búsqueda ISIS por ejemplo puede ser escrita dentro de un elemento de texto y los límites de búsqueda pueden ser tomados por medio de elementos de list check o radio box. Una vez que la búsqueda ha sido ingresada por el usuario en el campo, el valor será almacenado. El segundo componente es un botón que al pulsarse permitirá enviar los datos ingresados en los campos al servidor Web. Hay dos métodos básicos de transferir datos desde el cliente a la aplicación CGI. Estos son get y post.

En el método get el browser envía al servidor un formato string con la dirección URL (Uniform Resourse Locator) del cliente seguido por un caracter '?' y un string conteniendo los datos recolectados a través del form.
En el método post envía los datos al cgi a través de la entrada estándar la cual es leída por la aplicación cgi.
WWWISIS es capaz de leer y procesar ambos métodos siendo post recomendado para el trabajo. Cuando WWWISIS es ejecutado crea en memoria un registro del Archivo Maestro llamado CGI y un registro virtual de entorno, el cual puede ser accedido y formateado a través de parámetros cgi, cmd y prolog. Estos sirven para pasarle parámetros al WWWISIS. Los mismos son especificados en el archivo genparm.cgi.
WWWISIS cuenta con la posibilidad de incluir códigos html en el fomato para la visualización de los datos, usar literales, desarrollar una potente herramienta gráfica, e interfases de búsqueda cliente/servidor para bases ISIS.

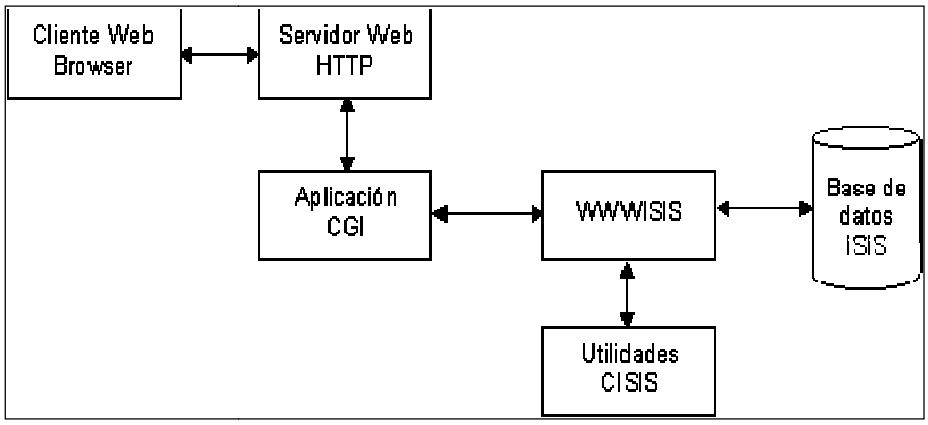


Figura 2.3: Arquitectura del Servidor Web con la herramienta WWWISIS

2.2.3 JAVAISIS

Es un software Cliente/Servidor en JAVA para consulta a base de datos CDS/ISIS en WEB. Su aspecto es similar a Winisis con la diferencia que permite la actualización y consulta de la base de datos vía una conexión remota usando el protocolo TCP/IP. La instalación es sencilla, además de no requerir mayor conocimientos que los necesarios para subir archivos por FTP. Fue concebido y desarrollado por Renato Enea (Florence, Italy), su distribución es gratuita y su versión actual es la 3.5 beta 1.

Es un programa Open Source por lo que se puede modificar y redistribuir de acuerdo a la licencia pública GNU (General Public Licence) y LGPL (Lesser Gerneral Public License. Al ser un software Open Source presenta una serie de ventajas para su implementación en bibliotecas. Se ofrece en forma gratuita y con su código fuente completo que permite corregir errores del software, modificarlo e integrarlo con otros programas.

El JAVAISIS está compuesto por dos programas uno servidor y el otro cliente. La interfase JAVAISIS Server es una aplicación java que se comunica con sus clientes a través de un puerto TCP/IP.

Como su instalación es simple se reduce al máximo el número de operaciones y configuraciones que se tiene que hacer para instalar la base de datos en línea. Se puede usar sobre diferentes plataformas (Windows 95, Windows 98, Windows NT, Windows 2000, Sun OS 5.5, Linux, HP-UX, IBM-AIX). JAVAISIS Cliente ha sido diseñado para ser visto y comportarse como sea posible al Winisis. Por lo tanto

usar JAVAISIS cliente no requiere mayor práctica o ningún esfuerzo de aprendizaje para los usuarios de Winisis o en general para los nuevos usuarios.

Las acciones que se pueden realizar los clientes en las bases de datos (como visualización, búsqueda o actualización) son definidos mediante parámetros en el servidor.

La bases de datos no puede ser actualizada por cualquier cliente, para realizar las actualizaciones es necesaria la utilización de contraseñas como opción básica pero necesaria de seguridad para proteger la consistencia de los datos.

La instalación es simple y es compatible con múltiples plataformas como Windows 95, Windows NT, Sun Solaris, Linux, HP-UX and IBM-AIX operating systems.

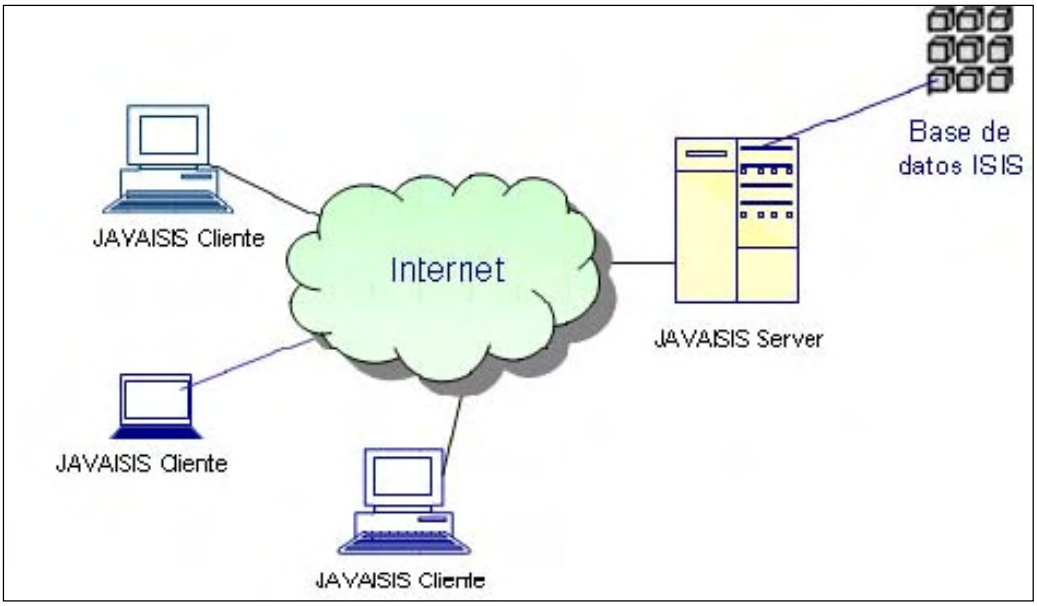


Figura 2.4: Arquitectura Cliente /Servidor del JAVAISIS

2.3 SERVICIOS WEB

2.3.1 DEFINICIÓN

Un servicio web (en inglés, *Web Service*) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de computadoras como Internet.

La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. ***(Consultar Referencia URL[24])***

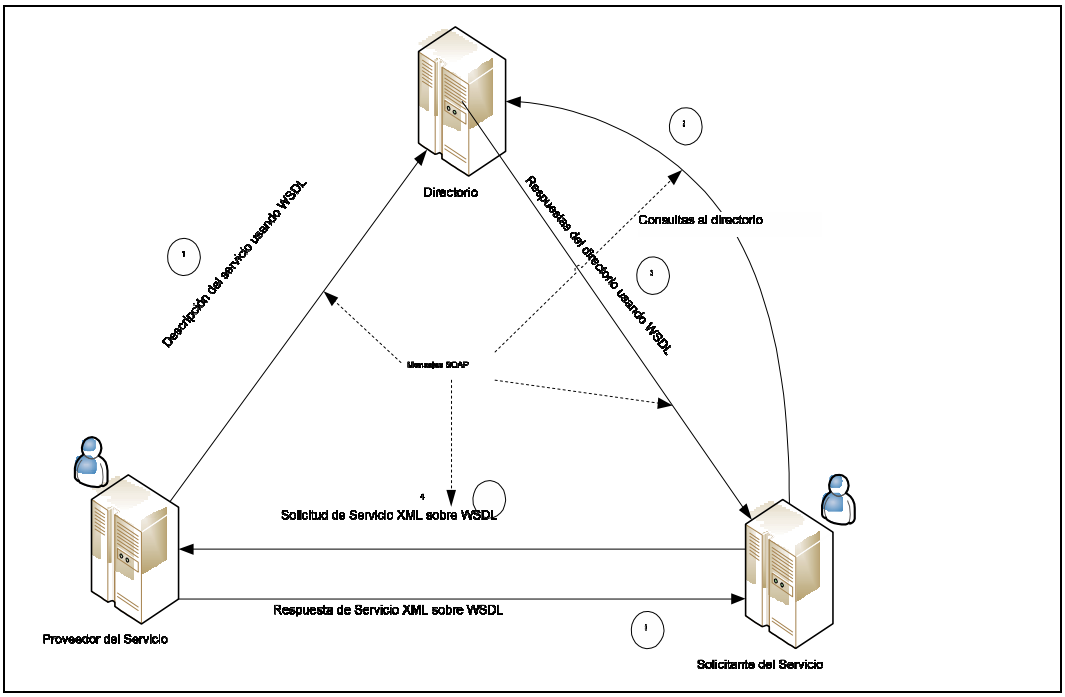


Figura 2.5: Modelo básico del funcionamiento de los Servicios Web

2.3.2 ESTÁNDARES EMPLEADOS

- Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- XML (Extensible Markup Language): Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call): Protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- WSDL (Web Services Description Language): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- UDDI (Universal Description, Discovery and Integration): Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.
- WS-Security (Web Service Security): Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados***. (Consultar Referencia URL[27])***

2.3.3 WEB SERVICES PROTOCOL STACK

Al conjunto de servicios y protocolos para los servicios web es conocido comúnmente como “Web Services Protocol Stack” y básicamente son utilizados para definir, localizar, implementar y hacer que un servicio web interactúe con otro. Este conjunto está conformado esencialmente de cuatro subconjuntos:

- Servicio de transporte
- Mensajería XML
- Descripción del servicio
- Descubrimiento de Servicios

✓ **Servicio De Transporte**

Es el encargado del transporte de los mensajes entre aplicaciones sobre la red. Incluye varios protocolos del nivel de aplicación. A continuación se relata sobre los más utilizados:

a) *HTTP (HyperText Transfer Protocol):*

Protocolo del nivel de aplicación más utilizado en la Internet. Es el protocolo que define la sintaxis y la semántica utilizada para la arquitectura web. En el contexto de los servicios web es utilizado para la transferencia de las transacciones XML a través de la red utilizando los mismos principios del HTML.

b) *FTP (File Transfer Protocol):*

Es un protocolo de la capa de aplicación encargado de los servicios de transmisión de archivos a través de redes soportadas sobre TCP. En el ámbito de los servicios web el FTP permite realizar modificaciones en equipos remotos evitando el uso de permisos sobre los archivos en la máquina cliente en sistemas operativos diferentes a Windows.

c) *SMTP (Simple Mail Transfer Protocol):*

Es un estándar de la capa de aplicación ampliamente utilizado para el envío de mensajes de correo electrónico a través de Internet. Es un estándar de Facto basado en texto, que requiere como cliente software de tipo POP3 o IMAP.

d) *BEEP (Block Exensible Exchange Protocol):*

Es un protocolo del nivel de aplicación , también conocido como BXXP, está diseñado para la interacción asíncrona punto a punto sobre una red TCP/IP Fue estandarizado por el IETF y provee un marco para administrar las conexiones punto a punto, autenticación., transporte de mensajes y manejo de errores.

e) *JMS (Java Message Service):*

Es una aplicación de interface de programación para JAVA (API) para el envío de mensajes entre dos o más clientes. Soportan dos modelos el modelo punto a punto y el modelo de publicación y suscripción. Una

aplicación JMS está compuesta por las siguientes partes:

- Un proveedor JMS que implementa las interfaces que proveen las características de administración y el control.
- Clientes JMS que son los componentes escritos en JAVA que producen y consumen los mensajes.
- Los Mensajes que son los objetos dato entre los clientes JMS.
- Objetos administradores que son objetos configurados previamente por un administrador del sistema para el uso de los clientes.
- Clientes Nativos que son programas que utilizan los mensajes de la API de manera similar que la API JMS.

✓ **Mensajería XML**

Es el conjunto encargado de la codificación de los mensajes en XML estándar y pueda así ser interpretado en cualquiera de los nodos de la red. Los componentes más utilizados en este conjunto son los siguientes:

a) *REST (Representational State Transfer):*

Fielding da la siguiente definición: "estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la World Wide Web". En resumen, es un conjunto de principios para el diseño de redes, que es utilizado comúnmente para definir una interfaz de transmisión sobre HTTP de manera análoga a como lo hace SOAP. Aunque REST como tal no es un estándar, posee un conjunto de estándares tales como HTML, URL, XML, GIF, JPG y tipos MIME.

Los principios de REST son:

- Escalabilidad de la interoperabilidad con los componentes.
- Generalidad de Interfaces.
- Puesta en funcionamiento independiente.
- Compatibilidad con componentes intermedios.

b) *RPC (Remote Procedure Calls):*

Es una tecnología de software que permite ejecutar una rutina en un equipo o segmento de red de manera remota. Es un paradigma popular para la implementación de sistemas distribuidos bajo arquitecturas cliente servidor.

c) *XML-RPC:*

Es un protocolo de llamada remota que utiliza XML como lenguaje de codificación y HTTP como mecanismo de transporte. Es un protocolo sencillo ya que solo define algunos tipos de datos y comandos.

Existen implementaciones de XML-RPC específicas para ActionScript, Delphi, C++, .NET, OClam, Common LISP, PHP y otros.

d) *XML (eXtended Markup Language):*

XML es uno de los lenguajes más utilizados para el intercambio de datos sobre la web. Su desarrollo se remonta en el año 1996 por el grupo de trabajo de la World Wide Web Consortium lanzando su primera versión el 10 de Febrero de 1998. El lenguaje XML está concebido para describir objetos de datos llamados Documentos XML y describir de cierta forma los programas que los procesan. Está restringido bajo la norma ISO 8879 el Estándar Generalizated Markup Language. Un documento XML es un objeto de datos que está bien formado, y se dice que lo está cuando tomado en su conjunto coincide con la producción del documento etiquetado, reúne todas las especificaciones de formato definidas y cada una de las entidades que se llaman directa o indirectamente están también bien definidas.

El XML es un lenguaje etiquetado, característica que le permite definir objetos de datos estructurados en partes bien definidas llamadas elementos. Una etiqueta es una señal realizada dentro del documento XML que delimita un segmento definido y con sentido de este documento.

Este es un ejemplo de XML:

```
<Edita_Mensaje>
  <Mensaje>
    <Remite>
      <Nombre>Nombre del remitente</Nombre>
      <Correo>Correo del remitente</Correo>
    </Remite>
    <Destinatario>
```

```
        <Nombre>Nombre del destinatario</Nombre>
        <Correo>Correo del destinatario</Correo>
      </Destinatario>
      <Text>
        <Asunto>
          Este es un documento sencillo sin atributos
        </Asunto>
        <Parrafo>
          Este es un documento sencillo
        </Parrafo>
      </Text>
    </Mensaje>
</Edita_Mensaje>
```

Adjunto al documento XML existe una definición de tipo de documento (DTD) en donde se describe la estructura y la definición de los datos de un documento XML. Una DTD describe normalmente los elementos, que son los segmentos etiquetados, la estructura, que es el orden en el cual van los elementos y el nivel de anidamiento.

```
<!ELEMENT Mensaje (Remite, Destinatario, Asunto, Text)*>
      <!ELEMENT Remite (Nombre, Correo)>
            <!ELEMENT Nombre (#PCDATA)>
            <!ELEMENT Correo (#PCDATA)>
      <!ELEMENT Destinatario (Nombre, Mail)>
            <!ELEMENT Nombre (#PCDATA)>
            <!ELEMENT Correo (#PCDATA)>
      <!ELEMENT Asunto (#PCDATA)>
      <!ELEMENT Text  (Parrafo)>
            <!ELEMENT Parrafo (#PCDATA)>
```

e) SOAP (Simple Object Access Protocol):

SOAP es un protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes de computadores. Básicamente es una vía de transmisión entre un SOAP Sender y un SOAP Receiver, pero los mensajes SOAP deben interactuar con un

conjunto de aplicaciones para que se pueda generar un “dialogo” a través de mensajes SOAP. Un mensaje SOAP es la unidad fundamental de una comunicación entre nodos SOAP. SOAP es básicamente un paradigma de una sola vía pero con la ayuda de las aplicaciones se puede llegar a crear patrones más complejos. SOAP básicamente está constituido por:

- Un marco que describe el contenido del mensaje e instrucciones de proceso.
- Un conjunto de reglas para representar los tipos de datos definidos.
- Convenciones para representar llamadas a procedimientos remotos y respuestas.
- Y algunos lineamientos entre SOAP y HTTP. ***(Consultar Referencia URL[20]) (Consultar Referencia URL[23])***

✓ **Descripción Del Servicio**

El servicio web debe contar con una interfaz pública la cual es descrita por un formato llamado WSDL (Web Services Descripción Language).

a) WSDL (Web Services Description Language)

WSDL es un tipo de documento XML que describe lo que hace un servicio web, donde se encuentra y la forma de ser invocado. Este provee información muy importante para los desarrolladores, este lenguaje describe el formato de los mensajes que utiliza y a cuales puede responder. Siempre un documento XML WSDL presenta los siguientes elementos:

- Tipos: Tipos de datos usados por los mensajes.
- Mensaje: Que datos son enviados desde un nodo a otro.
- Tipo de puerto: Define las operaciones que pueden ser llamadas.
  - Operación: Define la configuración de mensajes de entrada, salida y error.
  - Entrada: Mensaje que es enviado hacia el servidor.
  - Salida: Mensaje enviado hacia el cliente.

- Falta: Error en el envío de un mensaje.

- Límite: Es la descripción del protocolo que se está utilizando para transportar el mensaje que puede ser HTTP POST, HTTP GET, SOAP y MIME.
- Servicio: Define una colección de puertos (nodos); el puerto especifica una dirección para el límite definiendo así la comunicación para un nodo específico.

(Consultar Referencia URL[21])

✓ **Descubrimiento De Servicios**

UDDI (Universal Description Discovery and Integration): UDDI es un marco independiente de la plataforma para describir servicios, negocios e integrar servicios de negocios. La estructura de UDDI está basada sobre los servicios estándares de la web, lo que quiere decir que UDDI es accesible como otros servicios web. UDDI es un esfuerzo de la industria iniciada en septiembre de 2000 por Ariva, IBM, Microsoft y otras 33 compañías. Los propietarios de Servicios Web los publican en el registro UDDI. Una vez publicados se mantienen allí apuntadores a la descripción del Servicio Web y al servicio.

UDDI permite a los clientes buscar tal registro, encontrar el servicio deseado y extraer sus detalles. Estos detalles incluyen el punto de invocación así como otras características del servicio y su funcionalidad. La estructura de datos con UDDI está compuesta en cuatro partes:

- businessEntity
- businessService
- bindingTemplate
- tModel

businessEntity

Describe al proveedor del servicio web. Tiene datos como nombre de compañía, detalle de contacto y otra información del negocio.

businessService

Describe un conjunto lógico de uno o muchos servicios web.

bindingTemplate

Describe un único Servicio Web, describe toda la información técnica para que

el cliente pueda interactuar con él.

tModel

Representa especificaciones técnicas, metadatos sobre las especificaciones del documento, el nombre puntero URL, es presentado en forma de un documento WSDL. ***(Consultar Referencia URL [17])***

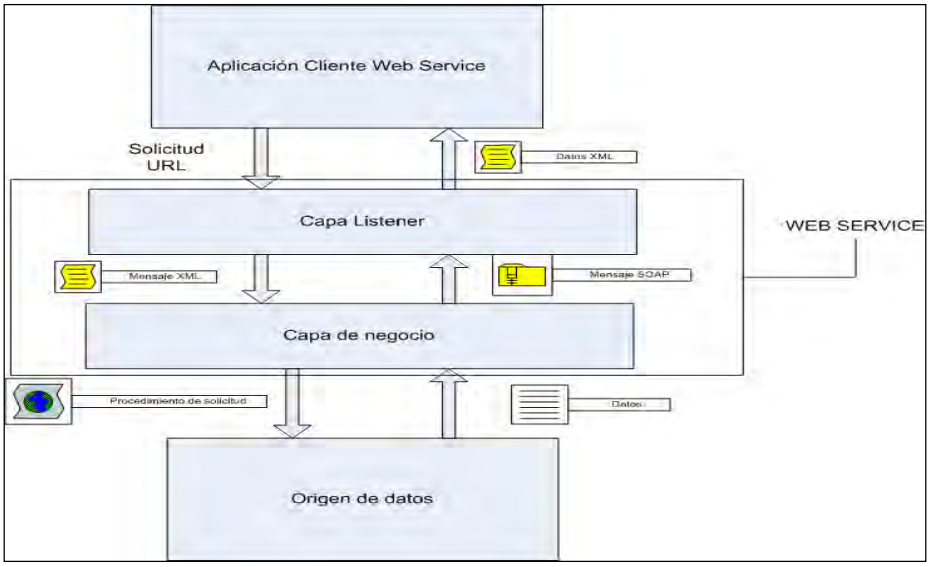


Figura 2.6: Funcionamiento de un Servicio Web

2.3.4 ESPECIFICACIONES ADICIONALES

Algunas especificaciones adicionales han sido desarrolladas o están empezando a ser desarrolladas con el ánimo de extender las capacidades de los servicios web. De manera habitual estas especificaciones son nombradas como ws-??. Algunas de las más importantes

WS-Security:

Es un protocolo de comunicaciones encargado de proveer seguridad a las aplicaciones de Servicios Web. Fue desarrollado originalmente por Microsoft, IBM, Verisign y Forum Systems, ahora el protocolo es llamado WSS. WSS ofrece mejoras en el sistema de mensajería SOAP para proveer calidad en la protección a través de la integridad de mensajes, confidencialidad y autenticación. Define como

usar encriptación XML y firma XML en SOAP, es una alternativa diferente a HTTPS.

WS-Reliability

Es un protocolo basado en SOAP para el intercambio de mensajes con distribución garantizada, sin duplicados y garantizando el orden del mensaje.

WS-RealiabilityMessaging

Esta especificación describe un protocolo que permite enviar mensajes SOAP de manera confiable entre sistemas distribuidos en presencia de fallas de los sistemas, componentes o aplicaciones. El objetivo de esta especificación es asegurar que el mensaje enviado por el emisor sea recibido por el receptor. La confiabilidad en los Servicios Web es algo difícil de definir, pero se puede realizar un símil de WSRM para XML como JMS para Java.

WS-Adressing

Esta es una especificación de mecanismos de transporte que permite a los Servicios Web comunicar información direccionada. Tiene una estructura compuesta en principio por dos partes. La primera, es una estructura para comunicar una referencia al nodo final del servicio web, y la segunda, es un conjunto de propiedades de direccionamiento con las cuales se asocia la información direccionada con un mensaje en particular. Las propiedades de direccionamiento son:

- Destinación de mensaje URI
- Origen del nodo final.
- Reenvío de nodo final.
- Falla del nodo final.
- Acción.
- Identificador único del mensaje.
- Relación con mensajes previos. WS-Transaction

Es una especificación desarrollada inicialmente por Microsoft, IBM y BEA Systems. Esta describe tipos de coordinación que son usadas con el marco extensible de coordinación descrito en la especificación WS-Coordination.

WS-Coordination

Es una especificación que describe un marco extensible para proveer protocolos que coordinen las acciones de aplicaciones distribuidas. El marco definido en tal

especificación habilita un servicio de aplicación para crear un contexto necesario para propagar una actividad a otros servicios y registrarlos a protocolos de coordinación. ***(Consultar Referencia URL [19])***

2.3.5 TECNOLOGÍAS ASOCIADAS

WSDL

WSDL es el acrónimo de Web Services Description Language (lenguaje de descripción de servicios Web). Se puede definir un archivo WSDL como un documento XML que describe un conjunto de mensajes SOAP y la forma en la que éstos se intercambian. En otras palabras, WSDL es a SOAP lo que IDL es a CORBA o a COM. WSDL es XML (es decir, se puede leer y modificar en gran parte de los casos) y es generado y utilizado por software.

El lenguaje de descripción de servicios web, Web Services Definition Language, (WSDL) nació en septiembre de 2000 de la mano de Microsoft, IBM y Ariba. Se basa en los lenguajes de definición NASSL44 (Network Accesible Services Specification), de IBM, y SCL45 (SOAP Contract Language) de Microsoft. En Marzo de 2001, estas compañías, con el apoyo de algunas otras, enviaron la versión WSDL 1.146 al W3C, donde fue publicado como una nota por lo que formalmente no es un estándar del W3C. ***(Consultar Referencia URL [18])***

La especificación de WSDL (1.1) no ha cambiado en absoluto desde su aparición, a pesar de haber un grupo de trabajo del W3C dedicado a WSDL que tiene borradores con diversas mejoras.

En esencia, WSDL es un contrato entre el proveedor del servicio y el cliente mediante el que el proveedor del servicio indica:

- Qué funciones que se pueden invocar
- Qué tipos de datos utilizan esas funciones
- Qué protocolo de transporte se utilizará para el envío y recepción de los mensajes (típicamente, pero no únicamente, mensajes SOAP).
- Cómo acceder a los servicios. En esencia, mediante qué URL se utilizan los servicios.

SOAP

SOAP (Simple Object Access Protocol), es un protocolo simple para el intercambio de información estructurada en un entorno distribuido y descentralizado. Utiliza XML para definir un framework extensible de mensajería proveyendo un formato de mensaje que puede ser intercambiado sobre una variedad de protocolos subyacentes. El framework fue diseñado para ser independiente de cualquier modelo de programación o cualquier semántica específica de alguna implementación. SOAP define el formato XML para mensajes.

Existen otras partes en la especificación SOAP que describen cómo representar los datos de un programa como XML y cómo utilizar SOAP para realizar llamadas a procedimiento remoto (RPC). Estas partes opcionales de la especificación se utilizan para implementar aplicaciones estilo RPC en las que el cliente envía un mensaje SOAP que contiene una función a la que se puede llamar, además de los parámetros para pasar a la función. El servidor, por su parte, devuelve un mensaje con los resultados de la función ejecutada.

Las aplicaciones SOAP con estilo de documento son muy flexibles. Muchos nuevos servicios XML Web Services aprovechan esta flexibilidad para diseñar servicios que sería difícil implementar mediante RPC.

La característica más notable de SOAP es que se ha implementado en diferentes plataformas de hardware y software, lo que significa que puede utilizarse para enlazar sistemas dispares dentro y fuera de una empresa. En el pasado se realizaron muchos intentos para conseguir un protocolo de comunicaciones común que pudiera usarse en la integración de sistemas. Sin embargo, ningún intento se ha generalizado tanto como SOAP. ***(Consultar Referencia URL [25])***

UDDI

UDDI, “Universal Description, Discovery and Integration”, es un elemento central del grupo de estándares involucrados en la tecnología servicios web. Define un método estándar para publicar y descubrir servicios en el contexto SOA. UDDI constituye las páginas amarillas de los servicios Web.

Una entrada en una lista UDDI es un archivo XML que describe un negocio y los servicios que ofrece. Los servicios se definen por un documento UDDI denominado Type Model o tModel. En muchos casos, tModel contiene un archivo WSDL que describe una interfaz SOAP a un servicio XML Web; tModel es suficientemente flexible para describir prácticamente cualquier tipo de servicio.

La especificación de UDDI nació casi a la vez que la de WSDL, de la mano de las mismas compañías, pero no ha llegado nunca al W3C. La versión actual es la 3.0, especificación que data de agosto de 2003, siendo gestionada por OASIS. La implementación de estas especificaciones se denomina "Registro UDDI", el cual proporciona un conjunto de servicios web de registro y consulta vía SOAP.

El propósito funcional de un registro UDDI es la representación de datos y metadatos acerca de servicios web. Tanto para ser usado en una red pública como dentro de la infraestructura interna de una organización, un registro UDDI ofrece un mecanismo basado en estándares para clasificar, catalogar y manejar servicios web de forma de que puedan ser descubiertos y consumidos por otras aplicaciones.
Varios registros UDDI se pueden agrupar para formar un UBR (UDDI Business Registry) con la idea de que se apliquen las mismas políticas de autenticación, cifrado, o se balancee la carga de trabajo.
Los UDDI y los UBR pueden ser públicos o privados. Los privados permiten el registro de los servicios web sólo a sus miembros, añadiendo, por lo general, ciertas medidas de seguridad. Sin embargo, permiten la consulta de sus registros por cualquier usuario. La mayor parte de los UDDI y UBR existentes son privados.
Las empresas que implementan registros UDDI facilitan en general herramientas gráficas (vía Web o locales) para interactuar con el registro así como APIs para integrar con las aplicaciones.

- **Estructuras de datos en UDDI**

  La información almacenada en un registro UDDI es un conjunto de las denominadas "estructuras de datos UDDI". Estas estructuras, en XML, definidas en la especificación, son las que el cliente intercambiará con el registro UDDI. Cada instancia de estas estructuras se identifica de manera única con un identificador denominado UUID (Universal Unique Identifier).

  Las principales estructuras (elementos XML) de alto nivel son las siguientes:

  - BusinessEntity. Contiene información básica de la empresa: persona de contacto, una clasificación de la empresa conforme a alguna de las taxonomías definidas, así como una descripción en lenguaje natural de las actividades de la empresa.
  - PublisherAssertion. Una empresa puede declarar su relación con otras empresas, por ejemplo, como socios o como clientes. Cada una de estas relaciones se modela como una PublisherAssertion.

- BusinessService. Este elemento muestra los servicios ofrecidos por una empresa. Estos servicios pueden ser servicios web o no. Una BusinessEntity se compone de uno o varios elementos businessService, y un elemento businessService puede ser usado por varios elementos BusinessService.
- BindingTemplate. Este elemento contiene referencias a descripciones técnicas (por ejemplo, URLs apuntando a manuales técnicos) y a las URL de acceso de los servicios web. Cada elemento BusinessService puede tener uno o varios elementos BindingTemplate.
- tModel. Este elemento describe la manera de interactuar con el servicio web y su comportamiento. Entre sus datos se encuentra la URL donde se encuentra el documento WSDL. También pueden aparecer aquí las categorías en las que se puede englobar el servicio (pueden ser distintas de las que podían aparecer en BusinessEntity). ***(Consultar Referencia URL[28])***

2.3.6 VENTAJAS DE LOS SERVICIOS WEB

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad *firewall* sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

2.3.7 INCONVENIENTES DE LOS SERVICIOS WEB

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (Common Object Request Broker Architecture).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI (Remote Method Invocation), CORBA o DCOM (Distributed Component Object Model). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en *firewall* cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

2.3.8 RAZONES PARA CREAR SERVICIOS WEB

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante *firewalls* que filtran y bloquean gran parte del tráfico de Internet, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran *ad hoc* y poco conocidas, tales como EDI (Electronic Data Interchange), RPC (Remote Procedure Call), u otras APIs.

Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada.
Se espera que para los próximos años mejoren la calidad y cantidad de servicios ofrecidos basados en los nuevos estándares. ***(Consultar Referencia URL [22])***

2.3.9 PLATAFORMAS

Servidores de aplicaciones para servicios Web:

- JBoss servidor de aplicaciones J2EE Open Source de Red Hat inc.
- Oracle Fusion Middleware
- IBM Lotus Domino a partir de la versión 7.0
- Axis y el servidor Jakarta Tomcat (de Apache)
- ColdFusion MX de Macromedia
- Java Web Services Development Pack (JWSDP) de Sun Microsystems (basado en Jakarta Tomcat)
- JOnAS (parte de *ObjectWeb* una iniciativa de código abierto)
- Microsoft .NET
- Novell exteNd (basado en la plataforma J2EE)
- WebLogic
- WebSphere
- Zope es un servidor de aplicaciones Web orientado a objetos desarrollado en el lenguaje de programación Python
- VERASTREAM de AttachmateWRQ para modernizar o integrar aplicaciones host IBM y VT

2.3.10 ASPECTOS SOBRE LA SEGURIDAD

La seguridad empezó a ser un problema en el momento en que SOAP se convirtió en un protocolo más generalizado que se ejecutaba en múltiples medios de transporte. Por ejemplo, HTTP proporciona diferentes modos de autenticar qué usuario realiza una llamada SOAP. Pero, ¿cómo se propaga esa identidad al dirigirse el mensaje desde HTTP a un transporte SMTP?

SOAP se diseñó como un protocolo de base; afortunadamente existen especificaciones en los textos para diseñar en SOAP que proporcionan más características de seguridad para servicios Web. WS-Security define sistemas de cifrado completos y WS-License define técnicas que protegen la identidad de la persona que llama y aseguran que sólo usuarios autorizados puedan utilizar un servicio Web.

2.3.11 DIFERENCIAS ENTRE SOA Y WEB SERVICES

La diferencia entre la tecnología de Servicios Web versus la arquitectura orientada a servicios: La diferencia radica a nivel de soporte de la infraestructura.

La infraestructura en este contexto referencia a la tecnología de ayuda y a los assemblies que soportan la implementación de una solución SOA. Los Web Services stand-alone requieren muy poco soporte de infraestructura. Sin embargo, de lo acuerdo a lo revisado, SOA requiere mucho soporte de infraestructura incluyendo muchas opciones de transporte múltiple, infraestructura de seguridad y soporte para mensajería responsable. Diferentes compañías, incluyendo Microsoft e IBM, están trabajando juntos en establecer especificaciones estándar que cubran el amplio rango de tecnologías de soporte para la infraestructura SOA. ***(Consultar Referencia URL[26])***

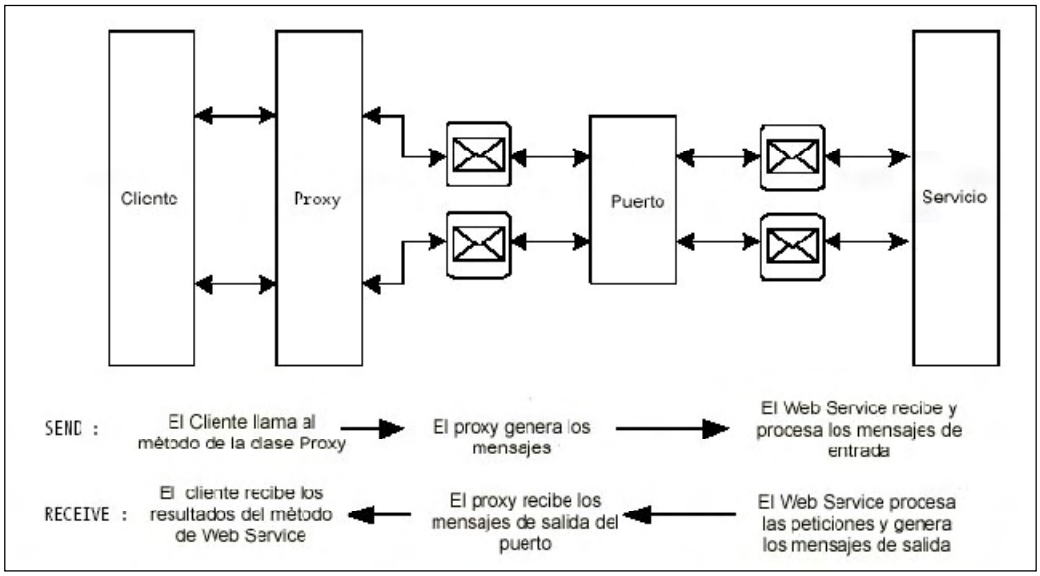


Figura 2.7: Arquitectura de Web Services mostrando la comunicación entre el cliente y el servicio.

2.4 ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

2.4.1 INTRODUCCIÓN

A lo largo de los años, en nuestras empresas se han ido acumulando una gran cantidad de aplicaciones distintas que se fueron desarrollando para tratar de resolver las distintas necesidades que iban surgiendo: ERPs, CRMs, Bases de datos, Mainframes, Sistemas CICS junto con aplicaciones WEB / J2EE, .NET, etc. Pero estas aplicaciones no se crearon con la idea de interactuar entre ellas, sino como herramientas para resolver un problema en un momento dado. En definitiva, el paisaje que nos encontramos es parecido a un archipiélago de aplicaciones.

La realidad del mercado nos ha llevado a las siguientes conclusiones:

- Nuestras aplicaciones internas están de alguna manera condenadas a entenderse, si queremos dar una respuesta ágil, a las necesidades de negocio que surgen.
- Las aplicaciones de nuestra empresa están también condenadas a entenderse de alguna manera, con las aplicaciones de las empresas con las que queremos cooperar para ofrecer mejor servicio a nuestros clientes.

Es, en este ámbito, donde SOA pretende hacer su aparición.
La palabra arquitectura en el mundo del software se podría definir como un conjunto de decisiones que hemos de tomar a la hora de organizar nuestras aplicaciones, como van a interactuar entre ellas, qué software se va a usar a la hora de comunicar entre ellas, protocolos, qué plataformas, máquinas, sistemas operativos, lenguajes de programación, qué tipo de mensajes se van a intercambiar, etc. Las decisiones que tomamos a la hora de decidir nuestra arquitectura son fundamentales, no tanto a corto plazo, sino más bien a largo plazo, y pueden ser a veces una trampa mortal.

La adopción de una arquitectura basada en servicios es actualmente una necesidad para la mayoría de las empresas. Esta necesidad no se justifica por una moda pasajera sino por la necesidad de lograr crear una infraestructura sólida que soporte la integración de los sistemas críticos de la empresa. Además, disponer de una infraestructura SOA también es un requerimiento para que una empresa pueda lograr una implementación exitosa de sistemas que permita alcanzar las metas que propone la teoría del BPM (Business Process Management, *Administración de*

Procesos de Negocio), lo que permite responder más rápido a las necesidades del negocio.

Las condiciones del mercado son más dinámicas que nunca. Esto significa que las organizaciones deben incrementar sus oportunidades lo más rápido posible y encontrar nuevas vías para satisfacer las cada vez más exigentes expectativas de los clientes. Más servicios, mejor adecuación, flexibilidad, mayores prestaciones en línea, tiempos de respuesta más cortos, fiabilidad, precisión. La lista de las necesidades de los clientes es interminable. Es importante saber si la organización puede dar respuesta a estas demandas y, en tal caso, conocer el camino que le permita distanciarse de la competencia. Una Arquitectura Orientada a Servicios (SOA) es la clave para dar respuesta a estas necesidades tanto ahora como en el futuro. En la figura siguiente podemos graficar cuales son las principales diferencia entre las arquitecturas clásicas y la orientación a servicios.

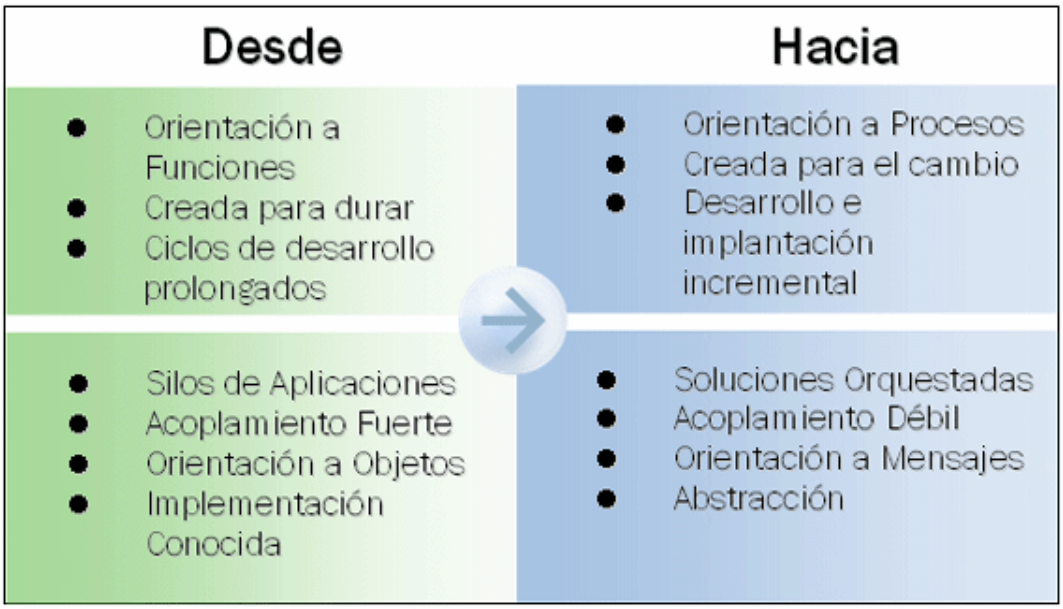


Figura 2.8 Evolución desde una arquitectura clásica hacia una arquitectura SOA

El número de empresas en las que la arquitectura SOA permite optimizar y agilizar los procesos de negocio es cada vez mayor. Al organizar las funciones de las aplicaciones centrales de su organización en servicios que comparten información y se basan en estándares, se conseguirán unos procesos mucho más eficientes y orientados al cliente.

Sin embargo, adoptar SOA no es una tarea sencilla, especialmente para aquellas organizaciones que no están acostumbradas a desarrollar sistemas distribuidos. Las principales compañías acometen la complejidad de sus

aplicaciones y entornos informáticos con la arquitectura orientada a servicios (SOA), que ayuda a desarrollar servicios empresariales modulares fáciles de integrar y reutilizar, construyendo una infraestructura informática verdaderamente flexible y adaptable. Con un enfoque de SOA, su organización informática dedicará más recursos y presupuesto a innovar y a prestar nuevos servicios empresariales.
(Consultar Referencia URL[10])

2.4.2 DEFINICIÓN

La mejor manera de comenzar a explicar SOA, es explicar qué NO es:

- *SOA no es un software*, no es un MOM (Middleware orientado a Mensajes), no es una EAI (Aplicación de integración empresarial), aunque una arquitectura SOA puede apoyarse en un conjunto de herramientas como MOMs o EAIs para conseguir su objetivo.
- *SOA no es una metodología de proyecto*, aunque a la hora de iniciar un proyecto orientado a conseguir una arquitectura SOA en una empresa, algunas metodologías se ajustan mejor que otras: es preferible seguir un modelo en iteraciones como las metodologías ágiles, que seguir metodologías Waterfall o Cascada
- *SOA no es otra forma de llamar a los Servicios Web*, aunque los Servicios Web son una herramienta válida para conseguir una arquitectura SOA, Pero, ¿Se puede implementar SOA sin usar Servicios Web? En teoría sí, pero en la práctica no. La gran ventaja de SOA es poder construir una misma arquitectura que incorpore tecnología y productos de diferentes proveedores, y por tanto la clave es la interoperabilidad. Construir un arquitectura software distribuida en n-capas y descompuesta en servicios débilmente acoplados con tecnología de un sólo fabricante no sirve de mucho, en realidad es un sobreesfuerzo para conseguir lo mismo que con una arquitectura tradicional. El beneficio que suele haber radica en reutilización y mantenimiento de las aplicaciones, pero desde luego muchas veces no compensa. La clave en las aplicaciones compuestas que reutilizan todos los servicios de una organización es la interoperabilidad y esta se consigue con los estándares conocidos como Servicios Web. Inicialmente existían 3: SOAP, WSDL y UDDI, pero a día de hoy la cosa se ha complicado, ya que las arquitecturas SOA corporativas necesitan de muchos más para ser interoperables a todos los niveles. Un caso habitual es la

interoperabilidad entre aplicaciones JAVA y .NET usando una misma arquitectura SOA

La Arquitectura Orientada a Servicios (SOA) es un modelo que interrelaciona unidades funcionales diferentes de una aplicación, denominado servicios, a través de interfaces y contratos bien definidos entre estos servicios. La interfaz se define de una manera neutral que debe ser independiente de la plataforma de hardware, del sistema operativo y del lenguaje de programación en los que se implemente el servicio. Esto permite que los servicios, construidos en una variedad de tales sistemas, interactúen entre sí de una manera uniforme y universal. ***(Consultar Referencia URL [6])***

Esta característica de tener una definición de interfaz neutral que no esté fuertemente sujeta a una implementación en particular se conoce como "*loose coupling"* o acoplamiento débil entre los servicios. que compone toda la aplicación. Por otra parte, *tight-coupling* significa que las interfaces entre los diferentes componentes de una aplicación están estrechamente interrelacionados en función y forma, haciendo que sean frágiles cuando se requiera alguna forma de cambio en partes de la aplicación o en toda la misma.

La necesidad de sistemas débilmente acoplados surgió asimismo de la necesidad de que las aplicaciones de negocios se tornaran más ágiles sobre la base de las necesidades de la empresa de adaptarse a su entorno cambiante, tales como políticas cambiantes, fuerzas de negocios, enfoques de negocios, asociaciones, posición en la industria y otros factores relacionados con los negocios que tienen influencia en la misma naturaleza de los mismos. Una empresa que puede actuar flexiblemente con respecto a su entorno como una empresa "*On Demand"*, es decir bajo demanda, donde hay cambios en la forma en que se realizan las cosas o el trabajo según sea necesario.

La arquitectura orientada a los servicios no es nueva, pero es un modelo alternativo con respecto a los modelos orientados a los objetos estrechamente acoplados de una manera más tradicional que han emergido en las últimas décadas. Al tiempo que los sistemas basados en SOA no excluyen el hecho de que se puedan construir servicios individuales con diseños orientados a los objetos, el diseño total está orientado a los servicios. Dado que permite objetos dentro del sistema, SOA puede estar basado en objetos, pero no está, en su totalidad, orientado a los objetos. La diferencia está en las interfaces propiamente dichas. Un

ejemplo clásico de un sistema proto-SOA que ha estado alrededor por un tiempo es *Common Object Request Broker Architecture (CORBA),* que define conceptos similares a SOA.

No obstante, el SOA actual es diferente en cuanto a que confía en un avance más reciente basado en *el eXtensible Markup Language (XML).* Al describir interfaces en un lenguaje basado en XML denominado *Web Services Definition Language (WSDL – Lenguaje de Definición de Servicios Web),* los servicios se han movido hacia un sistema de interfaz más dinámico y flexible que el *Interface Definition Language (IDL – Lenguaje de Definición de Interfaces)* anterior que se encontraba en CORBA. ***(Consultar Referencia URL [8])***

2.4.3 MITOS Y VERDADES SOBRE SOA

- ✓ Mitos
    - SOA es una Tecnología
    - SOA es nuevo y revolucionario
    -
    - SOA asegura que el negocio y IT trabajen juntos
    - SOA requiere de muchos consultores
    - Necesitamos construir SOA
- ✓ Verdades
    - SOA es una filosofía de diseño independiente de cualquier producto, tecnología o marca de mercado
    - SOA puede ser realizado vía Servicios Web, pero usando solo Servicios Web no necesariamente implica SOA
    - SOA no es una metodología
    - SOA debe ser incremental y desarrollado sobre las inversiones actuales
    - SOA es un medio, no un fin

2.4.4 TECNOLOGÍAS COMPONENTES DE SOA

En sí mismo SOA es un concepto abstracto sobre cómo se debe unir el software. Confía en las ideas y tecnologías más concretas implementadas en XML y en servicios de Web para existir en la forma de software. Asimismo, para funcionar con efectividad, también requiere soporte de seguridad, administración de políticas y una mensajería confiable. Esto puede ser mejorado aún más mediante la adición del procesamiento transaccional distribuido y la administración de estado del software distribuido.

La distinción entre los servicios de SOA y los servicios de Web reside en el diseño. El concepto de SOA no define exactamente cómo deberán interactuar específicamente, sólo cómo los servicios pueden comprenderse entre sí y cómo pueden interactuar. Es la diferencia entre definir una estrategia sobre cómo realizar un proceso, y la táctica sobre cómo realmente se hace. Por otra parte, los servicios de Web tienen pautas sobre cómo la mensajería entre los servicios necesita interactuar; es decir, es la implementación táctica de un modelo SOA que se ve más comúnmente en los mensajes de *SOAP* entregados sobre *HTTP*. De este modo, esencialmente los servicios de Web son un subconjunto específico acerca de cómo se puede implementar SOA.

Los Servicios Web son sólo un método para implementar SOA. Otros protocolos que también implementan directamente interfaces de servicio con WSDL y se comunican con mensajes de XML pueden estar asimismo comprendidos en SOA. Según lo investigado, ahora CORBA y los sistemas IBM MQ también pueden participar en SOA usando nuevos dispositivos que funcionan con WSDL. Si dos servicios necesitan intercambiar datos, también necesitan usar el mismo protocolo de mensajería pero las interfaces de datos permiten los mismos intercambios de información.

Para establecer un control apropiado de todo este intercambio de mensajes, así como también para aplicar las necesidades de seguridad, política y confiabilidad, hay un nuevo objeto de software que entra en la escena de SOA. Es el *Enterprise Service Bus (ESB),* que es responsable del control y del flujo apropiado, y tal vez también de las conversiones de todos los mensajes entre los servicios, usando cualquier cantidad de protocolos de mensajería posibles. No se requiere absolutamente el ESB, sobre todo cuando el proyecto es de envergadura

mediana y aun no se tiene un amplio portafolio de servicios, pero es un componente vital para administrar apropiadamente sus procesos de negocios en SOA. El ESB propiamente dicho puede ser un único motor o aun un sistema distribuido compuesto por muchos ESBs, todos funcionando juntos para mantener operativo el sistema SOA. Conceptualmente, ha evolucionado del mecanismo "store-and-forward" que se encontraba en los conceptos anteriores de la ciencia de la computación, tales como Message Queue y la computación transaccional distribuida.

Con respecto al desarrollador, las herramientas que usa necesitan conocer las capacidades de SOA y permitirle trabajar efectivamente con objetos de SOA. Esto incluye el proceso de diseñar el modelo, desarrollando servicios y objetos de servicio, y probar la aplicación de SOA en su totalidad. De este modo, las herramientas del desarrollador deberán estar listas para *Service-Oriented Application Design and Development (SOAD)* es decir Desarrollo de Aplicaciones Orientadas a Servicios, ejemplos de esto es la plataforma de desarrollo .NET de Microsoft. ***(Consultar Referencia URL [9])***

2.4.5 RELACIÓN DE SOA CON OTRAS TECNOLOGÍAS

SOA puede interactuar con una cantidad de otras tecnologías, pero con respecto a esto el encapsulado y el agregado de componentes tienen un rol significativo. Tal como se indicó anteriormente, un servicio de SOA puede ser un objeto simple, un objeto complejo, una colección de objetos, un proceso que contiene otros procesos, y asimismo una colección entera de aplicaciones que dan un único resultado. Fuera del servicio se lo ve como una única entidad, pero dentro de sí mismo puede tener *cualquier nivel de complejidad que sea necesario*.

SOA no es específico en lenguaje, excepto tal vez con respecto a XML y WSDL. Los servicios se pueden implementar en cualquier lenguaje de programación siempre y cuando pueda generar e interactuar con WSDL. SOAP en sí mismo no es un requisito absoluto, pero es un sistema de mensajería común. De este modo, los servicios miembros en SOA pueden ser implementados en casi cualquier variedad de lenguaje de programación y plataforma que dé soporte a WSDL.

Una aplicación basada en CORBA tiene muchos de los componentes necesarios para realizar interfaz con SOA. Si bien el Interface Description Language (IDL) en CORBA es conceptualmente similar a WSDL, no es exacto y por tanto necesita ser primero correlacionado con WSDL. Asimismo, los protocolos SOA de más alto nivel, tal como para la administración de procesos y políticas, necesitan ser usados en vez de los conceptos similares de CORBA. Debe tenerse en cuenta que esto es solo cuando un componente de CORBA (representado como un servicio) necesite interactuar con un servicio de SOA; dentro del modelo CORBA, todos los componentes de subconjuntos individuales pueden seguir funcionando como antes.

Los servicios de SOA son independientes de lenguaje de programación, pero los lenguajes Java y los nuevos de .NET están entre los principales lenguajes de desarrollo. La disponibilidad de interfaces bien definidas, así como también las abundantes implementaciones de estos lenguajes de varios protocolos, le dan a los desarrolladores una ventaja cuando se construye en este modelo. Aquí el lenguaje desempeña un rol en el desarrollo funcional de cada servicio, manipulando objetos de datos, y la interacción con otros objetos que estén lógicamente encapsulados dentro del servicio. ***(Consultar Referencia URL [7])***

2.4.6 PRINCIPIOS DE LA ORIENTACIÓN A SERVICIOS

Un problema con el que nos podemos encontrar a la hora de construir una aplicación SOA es si la aplicación construida realmente es una aplicación "SOA Compliant". Para comprobar si una aplicación lo es, la mejor forma de hacerlo es chequeando que la aplicación cumpla con los Principios de la Orientación a Servicios.

No existe una definición estándar de cuáles son los Principios de la Orientación a Servicios, por lo tanto, lo único que se puede proporcionar es un conjunto de Principios que estén muy asociados con la Orientación a Servicios. Estos Principios según Thomas Erl son:

- *"Los Servicios deben ser reutilizables":* Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.

- *"Los Servicios deben proporcionar un contrato formal"*: Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionalidades que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. De esta manera, todo consumidor del servicio, accederá a este mediante el contrato, logrando así la independencia entre el consumidor y la implementación del propio servicio. En el caso de los Servicios Web, esto se logrará mediante la definición de interfaces con WSDL.
- *"Los Servicios deben tener bajo acoplamiento"*: Es decir, que los servicios tienen que ser independientes los unos de los otros. Para lograr ese bajo acoplamiento, lo que se hará es que cada vez que se vaya a ejecutar un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama. Si conseguimos este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.
- *"Los Servicios deben permitir la composición"*: Todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel. En el caso de los Servicios Web, esto se logrará mediante el uso de los protocolos para orquestación (WS-BPEL) y coreografía (WS-CDL).
- *"Los Servicios deben de ser autónomos"* Todo Servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y nos podemos asegurar que así podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.
- *"Los Servicios no deben tener estado":* Un servicio no debe guardar ningún tipo de información. Esto es así porque una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución, es que un servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema de información sea del tipo que sea.
- *"Los Servicios deben poder ser descubiertos":* Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los Servicios Web, el descubrimiento se logrará publicando los interfaces de los servicios en registros UDDI.

Cuando se desarrollan aplicaciones SOA es muy útil y necesario tener en cuenta siempre estos principios, ya que nos van a dar las pautas necesarias para tomar ciertas decisiones de diseño complejas. Como se habrá podido observar, una característica muy importante de los Principios de la Orientación a Servicios, es que todos ellos se inter-relacionan

2.4.7 ELEMENTOS ESENCIALES DE SOA

Una aplicación SOA estará formada por un conjunto de procesos de negocio. A su vez esos procesos de negocio estarán compuestos por aquellos servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a buen término. Por último para ejecutar esas operaciones es necesario el envío de los datos necesarios mediante los correspondientes mensajes.

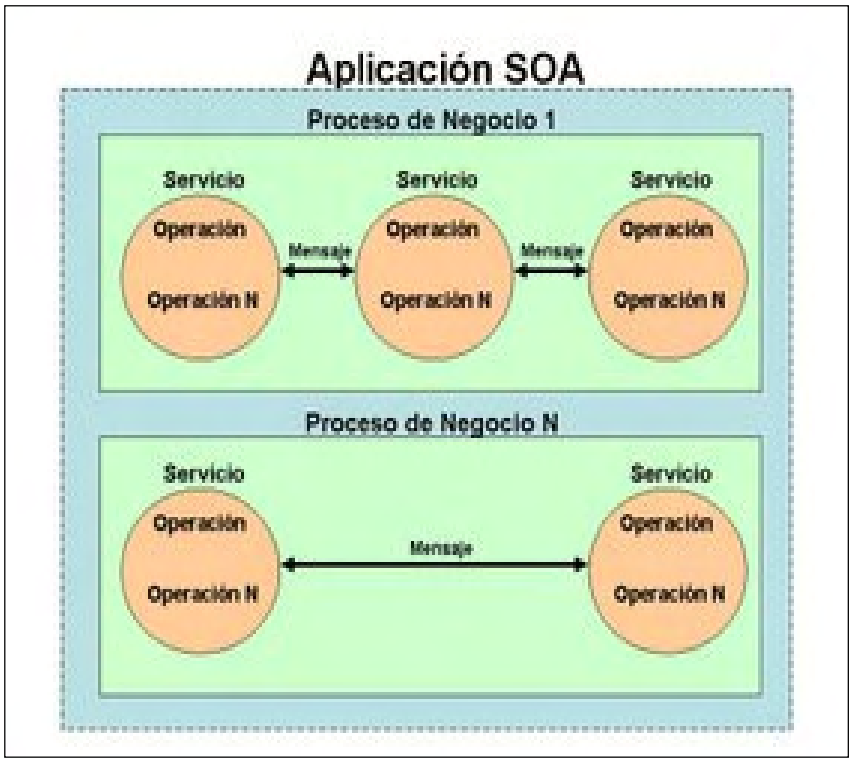


Figura 2.9: Ejemplo de aplicación SOA

Entre los elementos de SOA tenemos:

Proceso de negocio

Son un conjunto de operaciones ejecutadas en una determinada secuencia, intercambiando mensajes entre ellas con el objetivo de realizar una determinada tarea. También se define como proceso de negocio a un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas.

Las entradas son prerrequisitos que deben tenerse antes de que una función pueda ser aplicada. Cuando una función es aplicada a las entradas de un método, tendremos ciertas salidas resultantes.

Hay dos tipos de principales de procesos de negocio:

- Procesos Centrales: estos procesos dan el valor al cliente, son la parte principal del negocio
- Procesos de Soporte: dan soporte a los procesos centrales. Por ejemplo un proceso contable, un proceso de logística, etc.

Servicio

Es un contenedor de lógica, función sin estado, auto-contenida, que acepta una(s) llamada(s) y devuelve una(s) respuesta(s) mediante una interfaz bien definida. Los servicios pueden también ejecutar unidades discretas de trabajo como serían editar y procesar una transacción. Los servicios no dependen del estado de otras funciones o procesos. La tecnología concreta utilizada para prestar el servicio no es parte de esta definición.

Orquestación

Secuenciar los servicios y proveer la lógica adicional para procesar datos. No incluye la presentación de los datos.

Proveedor

Función que brinda un servicio en respuesta a una llamada o petición desde un consumidor.

Consumidor

Función que consume el resultado del servicio provisto por un proveedor; podrán ser tanto aplicaciones WEB, CRMs, ERPs, procesos Batch que se ejecutan de manera nocturna, etc.

Repositorio de servicios

Un repositorio de servicios será algún componente de la arquitectura SOA que permitirá tanto a los Consumidores como a otros servicios, descubrir que servicios existen, cual es su interfaz y donde se encuentran físicamente. Los objetivos de este componente serán:

- Crear un nivel de indirección para localizar a los servicios
- Servir como repositorio de información de los servicios existentes, contratos, calidad de los mismos, etc.

Bus de servicios (ESB)

Es el concepto que se refiere a la infraestructura de transporte de mensajes entre el motor de procesos y los servicios de los que dispone la empresa. Este componente de la arquitectura se utiliza sobre entornos heterogéneos de tecnologías de información. Tal como se ilustra en la siguiente figura:

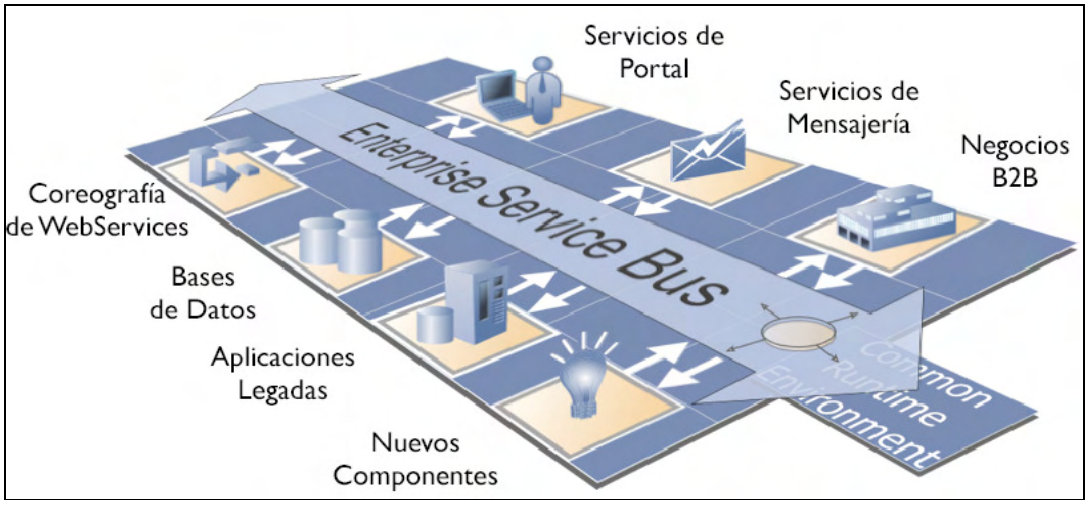


Figura 2.10 Elementos que interactúan con el Enterprise Service Bus

Este componente fundamental en la arquitectura SOA debe ofrecer:

- Conectividad entre frontales de aplicación y los servicios.
- Debe ser agnóstico de lenguajes de programación y tecnologías. Es decir debe ofrecer una forma de comunicación universal, para que todos puedan

entenderse (por ejemplo, puede usar XML como formato de comunicación de los mensajes)

- Debe ser capaz de ofrecer diferentes paradigmas de comunicación (sincronismo y asincronismo).
- Debería ser capaz de ofrecer otra serie de funcionalidades transversales como:
    - Trazabilidad de las operaciones (capacidad de "logging")
    - Mecanismos de seguridad (autenticación, autorización, encriptación, no repudio)
    - Mecanismos de transaccionalidad: protocolo de commit en dos fases (2PC) o transacciones en cadena y mecanismos de compensación, etc.
    - Enriquecimiento de mensajes, adaptación etc.
    - Control centralizado, mecanismos de monitorización.
    - Que incluyese un procesador de BPM, que permitiese construir servicios de mayor valor añadido en base a servicios básicos, simplemente definiendo la lógica en algún lenguaje (ej. BPEL)

Mensaje

Para poder ejecutar una determinada operación, es necesario un conjunto de datos de entrada. A su vez, una vez ejecutada la operación, esta devolverá un resultado. Los mensajes son los encargados de encapsular esos datos de entrada y de salida.

Operación

Es la unidad de trabajo o procesamiento en una arquitectura SOA.

2.4.8 TIPOS DE ARQUITECTURA SOA

SOA como estrategia en las empresas permite construir diferentes tipos de soluciones tecnológicas, estas se aplican dependiendo del escenario en el que se ubique la organización, y considerando que no son rígidas, es decir pueden combinarse para dar origen a una arquitectura customizada. Estas son:

Nivel 1: Escenario de Creación de Servicios *(Service Creation Scenario)*

Este tipo de escenario plantea la estructura básica de una arquitectura SOA (un proveedor de servicios, un consumidor de servicios y un directorio de publicación de los mismos), bajo este esquema se busca exponer funcionalidad de

los sistemas existentes o un nuevo sistema de servicios. Los servicios pueden ser consumidos por otros servicios u otras aplicaciones cliente dentro de la empresa y con otras empresas.

Este nivel se basa en la identificación, construcción e publicación de servicios web con un directorio de descubrimiento de los servicios es decir UDDI. Para esto la plataforma de los aplicativos comprometidos debe ser homogénea.

Nivel 2: Escenario de Conectividad de Servicios *(Service Conectivity Scenario)*

Este escenario plantea una arquitectura donde se tiene que integrar los servicios de los consumidores y de los proveedores, permitiendo el rehúso de funcionalidad existente o nueva a través de múltiples canales de comunicación.

Esta arquitectura es apropiada para las empresas que tienen un conjunto de servicios core (centrales) o sistemas que se desea estén disponibles como servicios para una gran variedad de clientes internos y externos. Plantea la posibilidad de interconectar procesos que interaccionan tanto hacia afuera como hacia dentro de la organización. Normalmente se asocia al concepto de ESB (*Enterprise Service Bus* o Bus de Servicio Empresarial) sobre todo si el numero de aplicativos y interconexiones se hace poco manejable y la plataformas tecnológicas de cada sistema son heterogéneas.

Nivel 3: Escenario de Interacción y Colaboración *(Interaction and Collaboration Scenario)*

La autenticación (single sign-on) y las funcionalidades basadas en roles (portales) son clave en este escenario para consolidar el acceso a la información y aplicaciones desde dentro de la empresa y entre empresas.

El uso de portales, documentos digitales, workflows y contenidos son los que priman en soluciones de este tipo. Ejemplos de esto son el uso de herramientas colaborativas que se enfocan en la *productividad* del personal y que pueden ser soluciones basadas en: Sharepoint de Microsoft, Oracle Portal o Websphere Portal

Nivel 4: Escenario de Administración de Procesos de Negocio (*Business Process Management Scenario BPM)*

La innovación y optimización a través de la implementación de estrategias de negocio vía el modelado, ensamble, despliegue y monitoreo de procesos a lo largo de todo el ciclo de vida es el que prima en este escenario.

BPM actúa como un habilitador para las empresas definiendo e implementando estrategias de negocio que se alinean a sus políticas, información, personal y tecnología, todo esto en tiempo real. Se enfoca en la mejora continua de procesos. Este tipo de arquitecturas es generalmente utilizado por grandes corporaciones y que bajo cuya dirección se encuentran empresas de distinto tipo de actividad pero que pertenecen a un mismo grupo empresarial y que comparten recursos, logística y procesos.

Nivel 5: Escenario de Información como Servicio *(Information as a Service Scenario)*

Este escenario plantea el uso de información resumida y fiable como servicios desde múltiples Fuentes de datos. Incluye aplicaciones como: data mining, BI. Se concentra en el conocer las necesidades y tendencias de mercado, es decir en el análisis de la información.

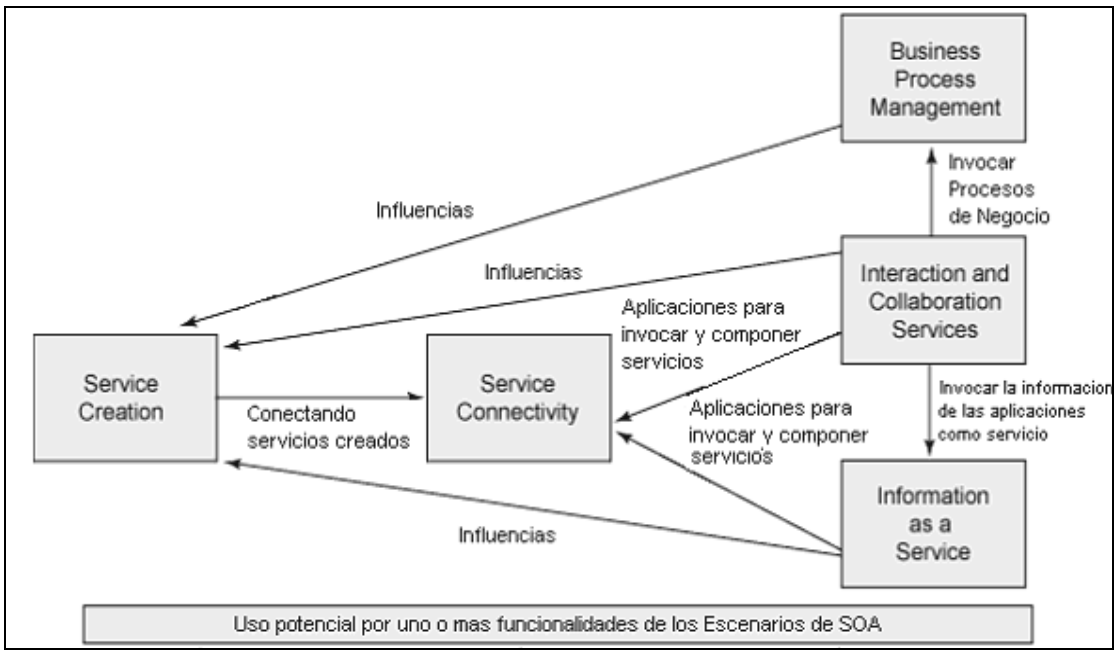


Figura 2.11 Escenarios de SOA y sus relaciones

Interrelacionando los escenarios SOA

Estos cinco escenarios no tienen por qué ser excluyentes entre sí, de hecho una solución SOA en algún momento y dependiendo de cuanto crezca en complejidad puede ir incorporando comportamientos propios de otro escenario o ser un escenario hibrido donde se puedan relacionar los comportamientos de cada uno, así en la figura anterior se muestra la potencialidad de la integración de funcionalidades de los escenarios SOA

El escenario de "*Service Creation"* (Nivel 1) esencialmente crea servicios. Los servicios se pueden identificar del modelo del proceso del negocio en el escenario de "*Business Process Management"* (Nivel 4). Los servicios identificados en el escenario de "*Information as a Service"* (Nivel 5) y el escenario de "*Interaction and Collaboration Service"* (Nivel 3) están siendo usados en el escenario de creación de servicios.

Los escenarios de *"Interaction and Collaboration"* (Nivel 3) mantienen el panorama de acceso que se basa en el acceso a los servicios de la empresa a través de su portal para invocar procesos de negocio del escenario de "*Business Process Management"* (Nivel 4); también se pueden invocar servicios definidos en el escenario de "*Information as a Service"* (Nivel 5).

Los servicios definidos por el escenario de *"Interaction and Collaboration"* (Nivel 3) y el escenario de *"Information as a Service"* (Nivel 5), usan el escenario de "*Service Connectivity"* (Nivel 2) para comunicarse entre ellos.

Los servicios creados en el escenario de "*Service Creation"* (Nivel 1) también usan el escenario de "*Service Connectivity"* (Nivel 2) para proveer conexiones entre los servicios creados de modo que juntos puedan formar servicios compuestos más complejos. ***(Consultar Referencia URL[14])***

2.4.9 CAPAS DE LA ARQUITECTURA SOA

En una arquitectura orientada a servicios la determinación de las capas que la componen es en general diversa y no existe un único modelo a seguir ya que cada solución se orienta hacia la solucionar a problemática de cada organización, pero si se indica que por lo menos debe contener:

- **Capa de Presentación**

Está formado por:

- Componentes de interfaz de usuario.
- Componentes de proceso de usuario.

- Independiente de implementación del resto de aplicación.
- Envía y recibe componente de entidades de negocio.
- Componentes de presentación:
    - No inician, participan, ni votan en transacciones
    - Obtiene una referencia al proceso actual del usuario si necesitan desplegar su data o actuar en su estado.
    - Pueden encapsular tanto la funcionalidades de visualización como de controlador

➢ **Capa de Negocio**

Está formado por:

- Interfaces de Servicio
    - Punto de entrada para abstraer implementaciones internas.
    - No debería cambiar cuando la implementación del componente de negocio cambia
    - Pueden existir distintas interfaces para la misma funcionalidad de negocio
    - Necesidad de interoperabilidad y rendimiento influye en el diseño de la interfaz de negocio
- Componentes de Negocio
    - Son invocados por la capa de presentación, interfaz de servicio, u otro proceso de negocio, usualmente en conjunto con alguna data de negocio para operar.
    - Proveen lógica de negocio o encapsulan otras lógicas de negocio
    - Son la raíz de las transacciones y votan en las transacciones donde participan
    - Validan operaciones de entrada y salida
    - Exponen operaciones de compensación
    - Pueden llamar a servicios externos a través de agentes de servicios, otros componentes de negocio e iniciar flujos propios de negocio
    - Pueden levantar excepciones hacia el llamador del servicio si existe un problema con las transacciones involucradas.
- Componentes de Proceso de Negocio
    - Maneja procesos que involucran múltiples pasos y transacciones largas

- Expone una interfaz que implementa un proceso que permite a la aplicación conversar con un servicio
- Utiliza solo componentes
- No necesita mantener estado conversacional mas allá de la actual actividad de negocio y la funcionalidad puede ser implementada como una transacción atómica única.
- Necesita encapsular funcionalidad y lógica reutilizable por muchos procesos.
- Lógica es intensiva o necesita a APIs y estructuras.
- Necesita control fino sobre flujo de datos y lógica.

➢ **Capa de Servicio**

Está formado por:

- Entidades de Negocio
  - Proveen acceso programático con estados a los datos del sistema.
  - Son usados como argumentos de los procesos de negocio
  - Son serializables
  - No acceden directamente a la base de datos
  - No manejan transacciones, eso lo hacen los procesos de negocio.
- Componentes de acceso a datos
  - Proveen los métodos de CREATE, READ, UPDATE y DELETE
  - Proveen método específicos para el motor de datos.
  - Encapsulan la complejidad del modelado de datos.
  - Estas son clases sin estado.
  - Típicamente invocan procedimientos almacenados.
  - Para generalizar funciones comunes usa herencia.
  - Para soportar diversos procesos de negocio usa como argumentos los métodos de las entidades de negocio.

(Consultar Referencia URL [11])

2.4.10 ¿QUIEN DEFINE LAS PAUTAS DE SOA?

Mucha gente se pregunta qué organismo es el encargado de estandarizar o por lo menos gestionar SOA. La respuesta es muy sencilla puesto que no hay ningún organismo que pueda hacerlo, ya que SOA es un concepto abstracto, el cual se rige única y exclusivamente por los principios de la Orientación a Servicios.

Pero entonces, ¿No hay manera de controlar la evolución de SOA?,¿Cada fabricante podrá hacer lo que quiera? Estas preguntas tienen una respuesta sencilla. Para comenzar, es necesario dejar un aspecto muy claro. Los Servicios Web, CORBA, MQSERIES, etc; son posibles tecnologías que se pueden utilizar a la hora de implementar una Arquitectura Orientada a Servicios y estas tecnologías sí que están estandarizadas y gestionadas por diversas organizaciones. Por lo tanto, las organizaciones que dirigen el rumbo de SOA, son aquellas que estandarizan las diferentes tecnologías utilizadas para implementar una Arquitectura Orientada a Servicios.

Por ejemplo, veamos el caso de los Servicios Web. Realmente los Servicios Web están formados por diferentes tecnologías, y por ello son varias las organizaciones que participan en su gestión. Concretamente son tres las organizaciones involucradas:

- **World Wide Web Consortium**

Organismo muy conocido porque es el encargado de la estandarización de HTML y XML (y tecnologías relacionadas). Referente a los Servicios Web, es el encargado de gestionar el protocolo de comunicación de los Servicios Web (SOAP), y el lenguaje de descripción de interfaces (WSDL). Más recientemente, W3C también se ha dedicado a estandarizar algunas de las extensiones WS-* de los Servicios Web. El W3C se caracteriza por ser muy formales y rigurosos a la hora de definir y gestionar tecnologías y protocolos, ofreciendo siempre las mejores garantías.

- **OASIS**

Anteriormente conocida como SGML Open, cambió su nombre para redirigir sus acciones de SGML hacia XML. Esta organización es bastante conocida por gestionar dos tecnologías muy conocidas. Es el encargado de desarrollar el estándar UDDI para el registro de Servicios Web, y también se encarga de gestionar la especificación ebXML es cual es un estándar para el intercambio de

datos entre aplicaciones B2B. Actualmente también se encarga de desarrollar extensiones WS-* para los Servicios Web. Estas extensiones son WS-BPEL creada para la orquestación de Servicios Web y WS-Security para todos los aspectos relacionados con la seguridad.

- **Web Services Interoperability**

Este organismo es reciente (apareció en 2002), y su principal objetivo es asegurar que se utilizan los estándares adecuados y no definirlos ni desarrollarlos. Por ello, este organismo ha definido un documento llamado perfil básico (Basic Profile), en el cual se indican cuales son los estándares que se deberían utilizar para diseñar arquitecturas inter-operables. Es decir, este documento es utilizado como mecanismo para generar arquitecturas SOA "compliant". Actualmente también se han preocupado por un aspecto tan importante como la seguridad, y han publicado un perfil de seguridad básico cuya finalidad es la misma que el anterior perfil, pero relacionado con la seguridad. Además esta organización ha prometido seguir publicando perfiles para distintos aspectos relevantes de las arquitecturas.

2.4.11 VENTAJAS Y DESVENTAJAS DE SOA

- ✓ **Ventajas**
  - Reducción de los periodos de desarrollo y los costes, los servicios SOA se reutilizan con facilidad y se ensamblan enseguida en nuevas aplicaciones compuestas.
  - Costes de mantenimiento más bajos, los servicios reutilizables reducen la cifra y la complejidad interna de los servicios informáticos.
  - Mayor calidad de servicios, la mayor reutilización de servicios devenga en una mayor calidad mediante múltiples ciclos de prueba por parte de distintos consumidores de servicios.
  - Costes de integración más bajos, los servicios normalizados saben cómo interactuar entre sí, lo que facilita la rápida conexión de aplicaciones distintas.
  - Menos riesgos, con menos servicios, pero reutilizables, se disfruta de mayor control sobre las políticas de responsabilidad informática y corporativa, así como disminuye el riesgo de incumplimiento global.

- Gestionar y controlar fácilmente los datos y aplicaciones críticos de la organización.
- Invertir la mayor productividad y velocidad conseguidas en beneficio del negocio y de las Tecnologías de Información.
- Conservar los valiosos sistemas heredados de la empresa.
- Agilizar la entrega de los servicios.
- Consolidar la complejidad subyacente a las TI y simplificar la integración permanente.
- Como arquitectura, propone un cambio a la empresa como un todo.
- El bajo acoplamiento de los servicios genera la independencia del uso de uno u otro servicio.
- A nivel operativo, por ejemplo, rapidez en la implementación de nuevos procesos, economía en el mantenimiento, agilidad en su diseño, minimizando los costos.
- No propone un cambio radical, o una revolución, sino que permite utilizar módulos previos, o sistemas antiguos haciendo la migración menos violenta.
- Es a la medida del cliente.
- Es absolutamente modular, más flexible, pero también tenemos la ventaja de que nos facilita la extensión de la estructura que existe hoy.

✓ **Desventajas**

- Requiere un cambio en las organizaciones, un alto esfuerzo.
- En la mayoría de las empresas, la proliferación de aplicaciones lleva al surgimiento espontáneo de una arquitectura corporativa “accidental”, esto es, no planeada.
- Incrementalmente se hace difícil y costoso el ser capaz de cumplir con los protocolos y hablar con un servicio a menos que se implemente un Bus de Servicio Empresarial.
- El conocimiento de los servicios es necesario para poder usar el servicio que proporciona un directorio de servicios. Dado que la Web es ilimitada por naturaleza, es imposible mantener todos los servicios web en un único directorio.
- Implica conocer los procesos del negocio, clasificarlos, extraer las funciones que son comunes a ellos, estandarizarlas y formar con ellas

capas de servicios que serán requeridas por cualquier proceso de negocio.

Al 2008, SOA será predominante en las prácticas de Ingeniería de Software, terminando con 10 años de dominio de las arquitecturas monolíticas. ***(Consultar Referencia URL [15])***

2.4.12 ELEMENTOS DE SOA QUE SON IMPORTANTES PARA SU ÉXITO

Algunos de los puntos a tener en cuenta son los siguientes:

- Como primer punto se encuentra la flexibilidad. SOA es la primera arquitectura de Tecnologías de Información que asume lo que los negocios han sabido desde hace mucho tiempo. Se trata esencialmente de un conjunto de servicios sueltos, donde cada uno es relativamente económico para construirlo o reemplazarlo si es necesario. Al ser independientes, el poder unirlos permite a SOA adaptar cambios, cuestión imposible para arquitecturas tradicionales.
- En la Arquitectura Orientada a Servicios, se puede reemplazar un servicio sin tener que preocuparse por la tecnología fundamental; la interfaz es lo que importa, y está definida en un estándar universal en servicios Web y XML. Esto es flexibilidad a través de la interoperabilidad. También es la habilidad de asegurar los activos existentes, aplicaciones y bases de datos legales y hacerlos parte de las soluciones empresariales extendiéndolos al SOA en vez de reemplazarlos. El resultado en la red es la habilidad de evolucionar rápida y eficientemente, en otras palabras, adaptarse "orgánicamente" de acuerdo a la demanda del negocio. Esto es realmente nuevo.

- En segundo lugar está la relevancia para el negocio. SOA es tecnología expresada a un nivel que tiene un significado importante para la colaboración del negocio y profesionales del área. Sus servicios actuales pueden coordinar unidades de trabajo muy cercanas a las actividades del negocio; piense, por ejemplo, en un servicio llamado "Actualización de órdenes de trabajo". Éstos son inmediatamente relevantes para los analistas de la empresa que participan en la creación y definición de nuevos procesos permitiendo el "Servicio Dirigido Empresarial".

- Los negocios y las TI se enfocan en la lógica del negocio y la comunicación; finalmente comparten el lenguaje de servicios. Esto también es relativamente nuevo y tendrá implicaciones en la entrega de servicios. ***(Consultar Referencia URL[16])***

2.4.13 CASOS DE NEGOCIO PARA SOA

Los casos de negocio que son fuentes comunes de proyectos en la industria, para utilizar una arquitectura orientada a servicios son los siguientes:

- Compra, adquisición o fusión de empresas en la industria, en las cuales es requerido integrar uno o varios de los sistemas empresariales.
- Automatización de procesos de negocio en las empresas. Los procesos de negocio pueden abarcan todo el ecosistema de empresas participantes en un proceso, tales como: clientes, empresas proveedoras, aliados de negocios, proveedores de servicios de públicos, entidades reguladoras, bancos, centrales de riesgo entre otros.
- Creación de una nueva línea o unidad de negocios en una empresa. Este caso normalmente requiere que la unidad de negocios soporte parte de sus operaciones con sistemas de información, sin embargo, estos sistemas comúnmente requieren integrarse con otros sistemas para intercambio de información y servicios, para lo cual se puede utilizar una arquitectura SOA.
- Modernización de Mainframes o Sistemas con tecnologías legadas. Este caso típico requiere que las empresas incrementalmente modernizar los sistemas legados. Para este caso, se utilizan normalmente integradores que permiten exponer servicios de los Mainframes o sistemas legados, de tal forma, que cuando las aplicaciones nuevas de las empresas requieran datos o transacciones de los sistemas legados, lo realicen a través de las tecnologías estándares propuestas por SOA, evitando el acoplamiento de los sistemas modernos con los sistemas antiguos. A futuro, se puede establecer un plan incremental que permita reemplazar módulos funcionales del Mainframe, cambiando la implementación tecnológica, sin cambiar la interfaz de servicios.
- Inteligencia de negocios. Un caso muy común en las empresas es consolidar la información de diferentes sistemas y crear un repositorio sobre el cual se puedan aplicar herramientas de inteligencia de negocios para mejorar los procesos de toma de decisiones. La extracción, enriquecimiento y homologación de información de los diferentes sistemas empresariales, se

puede realizar utilizando una arquitectura SOA, en la cual normalmente se utilizan: Buses Empresariales de Servicios (ESB) y adaptadores de integración.

- Consolidación de Información de Clientes.
- Creación de una nueva línea o unidad de negocios en una empresa.

2.4.14 BARRERAS A VENCER PARA OBTENER EL ÉXITO DE SOA

SOA es un nuevo horizonte para las TI. Como cualquier gran cambio, las principales barreras son organizacionales, no técnicas. A continuación ejemplificaremos algunas:

- Administración: Servicios compartidos es lo principal para utilizar SOA. La habilidad para ensamblar rápidamente aplicaciones o procesos está basada en la disponibilidad de algunos servicios que pueden ser compartidos. Hacer esto, por definición, requiere administración.
- Desarrollo Cultural: Al utilizar SOA se requiere un cambio significativo en el estilo de programar. Muchos desarrolladores utilizan equipos diferentes para resolver problemas de manera independiente para cada aplicación. En SOA necesitarán escribir aplicaciones para ser re-utilizadas en mente, usando códigos existentes, a los cuales se podrá tener acceso constantemente.

2.4.15 LO QUE PUEDE LOGRARSE CON UNA ARQUITECTURA SOA

SOA surge de la necesidad de hacer que los sistemas de negocios de IT sean más ágiles con respecto a los cambios en la empresa. Al permitir relaciones fuertemente definidas, si bien implementaciones específicas flexibles, los sistemas pueden obtener las ventajas de los otros sistemas existentes y, no obstante, estar lista para cambios futuros en sus interacciones.

Para dar un ejemplo específico, una organización minorista de indumentaria que posee una cadena internacional de 500 tiendas necesita cambiar frecuentemente sus diseños para mantenerlos a la moda. Esto podría significar no sólo cambios en estilos y colores, sino también en materiales, fabricación y entrega. Cambiar de un proveedor a otro puede ser un proceso de software complicado si los sistemas que hay entre el minorista y el fabricante son incompatibles. La flexibilidad de una interfaz de WSDL para las operaciones puede permitir que cada compañía mantenga sus sistemas existentes tal como son. En vez de ello, pueden sólo hacer coincidir las interfaces de WSDL y establecer nuevos convenios de nivel

de servicio en vez de reconstruir totalmente sus aplicaciones de software. Este es un cambio horizontal para la empresa, es decir, están cambiando asociados al tiempo que esencialmente todas las operaciones de negocios permanecen siendo las mismas en su mayoría. Aquí las interfaces de negocios pueden cambiar de una manera mínima y las operaciones internas pueden permanecer sin cambios, al tiempo que las interfaces de negocios todavía pueden funcionar juntas externamente.

Otra forma es el cambio interno, en la que la organización minorista ahora decide que también alquilará espacio dentro de la cadena de tiendas minoristas a vendedores de boutiques, tal como el modelo de negocios de tienda a tienda. Aquí la mayoría de las operaciones de negocios de la compañía permanecen sin cambios, pero ahora se necesita un nuevo software interno para manejar ese convenio de alquiler. Internamente, es posible que el sistema de software sea objeto de una revisión, pero necesita hacer esto sin afectar seriamente las interacciones con los sistemas existentes de sus proveedores. El modelo SOA en este caso permanece intacto, al tiempo que la implementación interna cambia. Se pueden agregar nuevos aspectos al modelo SOA para agregar las nuevas responsabilidades de los acuerdos de alquiler, mientras que el sistema regular de administración minorista continúa siendo el usual.

Para continuar aún más con la idea del cambio interno, puede que el gerente de tecnología, encuentre que se puede usar la nueva configuración de otras maneras, tal como rentar también espacio de “poster” para publicidad. Aquí surge una nueva propuesta de negocios del modelo SOA flexible re aplicado en un nuevo diseño. Este es un nuevo resultado de un modelo SOA y una nueva oportunidad que podría no ser posible anteriormente.

También son posibles los cambios verticales, donde el minorista pasa totalmente de vender sus propias ropas a alquilar exclusivamente espacio a través del modelo tienda-en-tienda. Si se toma completamente desde una base cero, un cambio vertical comprendería una reestructuración significativa del modelo SOA, tal vez con nuevos sistemas, software, procesos y relaciones. La ventaja del modelo SOA en este caso es que funciona desde la perspectiva de las operaciones y de los procesos de negocios, en vez de aplicaciones programas, permitiéndole a la administración de los negocios identificar claramente que necesidades deben ser agregadas, modificadas o eliminadas sobre la base del funcionamiento de la empresa. Luego se pueden estructurar los sistemas de software para ajustar los

procesos de negocios, en vez de ser lo opuesto, según se ve comúnmente en muchas plataformas de software.

Tal como se puede ver, el cambio y la capacidad del sistema SOA para adaptar a él son los elementos más importantes aquí. Para los desarrolladores, tales cambios pueden ocurrir dentro de su contexto de trabajo o fuera de él, según si hubiera cambios que son necesarios en cuanto a cómo las interfaces son definidas y cómo interactúan entre sí. En vez de ser el rol del desarrollador, es el del arquitecto engendrar la mayoría de los cambios en un modelo SOA. Esta división de trabajo, por la cual el desarrollador se concentra en crear unidades funcionales definidas como servicios, y el arquitecto y el modelador se concentran en la forma en que las unidades se ajustan entre sí, ha existido por más de una década, comúnmente representado en el *Universal Modeling Language* (UML) Lenguaje Universal de Modelado. ***(Consultar Referencia URL [16])***

2.4.16 COMO PUEDE BENEFICIAR SOA A LOS NEGOCIOS

El establecimiento de una arquitectura orientada a servicios puede ayudar a preparar tanto a las tecnologías de información, como a los procesos de negocios para un cambio rápido. Aún en las etapas tempranas de adopción de una SOA, su organización puede beneficiarse.

- Aumenta los ingresos, crea nuevos caminos al mercado; crea nuevos valores a partir de sistemas existentes.
- Provee un modelo de negocios flexible, busca el reaccionar más rápidamente a los cambios del mercado.
- Reduce a mediano plazo los costos, elimina los sistemas duplicados; construye una vez y potencia; mejore el tiempo de salida al mercado.
- Reduce los riesgos y la exposición, mejora la visibilidad en las operaciones de negocio.

El enfoque SOA puede colocar un puente entre lo que usted quiere que su negocio cumpla y las herramientas de infraestructura que tiene que tener para llegar allí.

- Disminuya los tiempos de los ciclos de desarrollo e implementación usando bloques de construcción de servicios reutilizables, prefabricados.
- Integre por toda la empresa, incluso los sistemas históricamente separados y facilite las fusiones y adquisiciones de empresas.

- Reduzca tiempos de ciclo y costos pasando de transacciones manuales a automáticas.
- Facilite la realización de negocios con los asociados de negocios aumentando su flexibilidad.

2.4.17 CÓMO SOA PUEDE AFECTAR LOS RESULTADOS DE LOS NEGOCIOS

Dependiendo de las prioridades de su industria, SOA puede ayudar a reducir el tiempo y los costos de entrega de nuevos servicios. Ayude a su empresa a responder más rápido a las exigencias del cliente. Proporcione a los clientes una experiencia de usuario unificada. Fortalezca los recursos de seguridad y reduzca los riesgos.

- Problema: La industria automotriz enfrenta problemas de calidad, los costos de garantías promedian US$ 700 por vehículo en los Estados Unidos.
    - Solución: Trabajar hacia multi-proveedor en sistemas embarcados e integración de software con SOA.
- Problema: Las empresas de atención de salud deben abordar la subida de costos, tiempos de respuesta lentos y la calidad inconsistente de los registros de pacientes.
    - Solución: Usar SOA para integrar a los pagadores, proveedores y hospitales.
- Problema: La industria electrónica está cambiando de la fabricación tradicional a la configuración bajo pedido.
    - Solución: Construir SOA que facilite la producción en masa con personalización de último minuto.
- Problema: La industria de bancos debe tratar con silos de información, redundancia y reutilización de datos, mientras está bajo una presión constante para crecer.
    - Solución: Responder con SOA para acelerar el desarrollo y la entrega de nuevos productos y servicios.
- Problema: Los minoristas se enfrentan al crecimiento exponencial de los datos (por ejemplo, RFID) que no son potenciados eficazmente.
    - Solución: Entregar información casi en tiempo real para optimizar la cadena de abastecimiento con SOA.

- Problema: Las empresas de telecomunicaciones se enfrentan con “islas” de infraestructuras, múltiples sistemas legados y entornos heterogéneos.
    - Solución: Diseñar SOA que ofrezca una única vista del cliente, desde la activación de cuenta al autoservicio, facturación y atención al cliente.

      (Consultar Referencia URL [14])

2.5 ENTERPRISE APPLICATION INTEGRATION (EAI)

2.5.1 DEFINICIÓN

Enterprise Application Integration (EAI) o Integración de Aplicaciones de Empresa se define como el uso de software y principios de arquitectura de sistemas para integrar un conjunto de aplicaciones. ***(Consultar Referencia URL [12])***

2.5.2 ORIGEN

Las compañías están constantemente implementando nuevas soluciones de manera informal, tanto al nivel de negocio como técnico. Esto ha provocado que se formen sistemas aislados.
Debido al aumento de la necesidad de comunicación e intercambio de datos entre estas aplicaciones independientes se ha desarrollado una disciplina cuyo objetivo es lograr la comunicación entre todos los sistemas que operan en una empresa, la integración de aplicaciones empresariales (EAI).

2.5.3 JUSTIFICACIÓN DE LA EAI

EAI es el proceso de conectar las aplicaciones con otras para intercambiar información . Cuando dichos sistemas no pueden compartir su información efectivamente se crean cuellos de botella que requieren de la intervención humana en la forma de toma de decisiones o en el ingreso mismo de la información.

Durante varias generaciones, los sistemas de las empresas han servido para un propósito especifico a un único usuario o grupo de usuarios, los cuales actúan como la interfaz de dicho sistema con el resto de la organización, con lo cual se ha limitado su conexión con otros sistemas modernos o más amplios en la empresa, más aún por la creciente demanda de las empresas por compartir datos y usarlos en sus procesos sin tener que realizar cambios en sus aplicaciones o en sus estructuras de datos.

Uno de los retos que encaran las organizaciones modernas es darles a sus empleados información completa en tiempo real. Muchas de las aplicaciones en uso

actualmente se apoyan en tecnologías antiguas, por lo cual esos sistemas enfrentan dificultades a la hora de mover esta información entre las aplicaciones.
EAI, como una disciplina, busca solventar muchos de esos problemas, así como crear nuevos paradigmas para ciertamente mejorar las organizaciones, tratando de trascender el objetivo de conectar las aplicaciones individuales para buscar ser un mecanismo de incrementar el conocimiento de la organización y crear ventajas competitivas futuras a la empresa. ***(Consultar Referencia URL [12])***

2.5.4 OBJETIVO DE LA EAI

EAI puede ser usado con diferentes fines:

- Integración de datos (información): asegurando que la información en varios sistemas sea consistente. Esto también se conoce como EII (Enterprise Information Integration).
- Integración de procesos: enlace de los procesos de negocios entre diferentes aplicaciones.
- Independencia de proveedor: extrayendo las políticas o reglas del negocio de las aplicaciones e implementándolas en un sistema EAI, de forma que cualquiera de las aplicaciones usadas pueda ser cambiada sin que dichas reglas de negocio deban ser reimplementadas.
- Facade común: Un sistema EAI puede actuar como el front-end de un cúmulo de aplicaciones, proporcionando una interfaz de acceso única y consistente a esas aplicaciones y aislando a los usuarios sobre la interacción con distintas aplicaciones.

2.5.5 PATRONES DE EAI

Patrones de Integración

Hay dos patrones que implementan los sistemas de EAI:

- Mediación: aquí, los sistemas de EAI actúan como el vínculo de los enrutadores entre varias aplicaciones. En el lugar en el cual ocurre un evento interesante en alguna aplicación (e.j., se crea una nueva información, se completa una nueva transacción, etc.) se notifica a un módulo de integración del sistema EAI. El módulo entonces propaga esos cambios a las otras aplicaciones relevantes.

- Federación: en este caso, el sistema EAI actúa como un consolidador de información entre varias aplicaciones. Todos los accesos del 'exterior' a cualquiera de las aplicaciones son recibidos por el sistema EAI, y éste está configurado para exponer sólo la información relevante conectándose a las aplicaciones del mundo exterior, y efectuar todas las interacciones con las aplicaciones internas sin intervención del agente externo.

Ambos patrones son usados en conjunto frecuentemente. El mismo sistema EAI puede tener varias aplicaciones en sync (mediación), mientras sirve requerimientos de agentes externos contra esas aplicaciones (federación).

Patrones de Acceso

EAI soporta patrones de acceso tanto síncronos como asíncronos, el primero es el habitual en el caso del patrón de mediación y el segundo en el caso de federación.

Vida de los patrones

Una operación de integración puede ser de "corta vida" (por ejemplo, puede mantenerse la sincronía de los datos entre dos aplicaciones en un segundo) o de "larga vida" (por ejemplo, en uno de los pasos puede ser necesario que el sistema EAI requiera de la aprobación por parte de un agente humano de un préstamo y que éste necesite horas o días para autorizarse).

2.5.6 TOPOLOGÍAS DE EAI

Hay dos topologías principales: hub-and-spoke, y bus. Cada una de ellas tiene sus propias ventajas y desventajas:
En el modelo hub-and-spoke, el sistema EAI actúa como el centro (el concentrador), el cual interactúa con las aplicaciones vía las conversaciones (o spokes).
En el modelo de bus, el sistema EAI es el bus (o es implementado como un módulo residente en un bus de mensajes existente o un un middleware orientado a mensajes).

2.5.7 TECNOLOGÍAS

Múltiples tecnologías son usadas para implementar cada uno de los componentes de un sistema EAI.

- Bus/hub: este se implementa frecuentemente al ampliar la funcionalidad de productos middleware existentes (servidores de aplicaciones, buses de mensajes) o se implementa como un programa monolítico (ej., sin usar ningún middleware), que actúa como su propio middleware.
- Conectividad de aplicaciones: el bus/hub se conecta a las aplicaciones mediante un conjunto de adaptadores (también conocidos como conectores). Esos son programas que conocen como interactuar con la aplicación específica. El adaptador efectúa una comunicación en dos vías, enviando requerimientos del hub hacia la aplicación, y notificando al hub cuando un evento de interés ocurren en la aplicación (un nuevo registro es insertado, una transacción es completada, etc.). Los adaptadores pueden ser tanto específicos a la aplicación o a un conjunto de aplicación. El adaptador puede residir en el mismo espacio de procesos que el bus/hub o ejecutarse en una localización remota e interactuar con el hub/bus a través de protocolos estándares de industria como colas de mensajes, servicios web, o protocolos propietarios.
- Formateo de datos y transformación: para prevenir que cada adaptador tenga que convertir los datos que van o vienen de otras aplicaciones, los sistemas EAI usualmente emplean un formato de datos común, al cual y desde el cual se convierten los formatos de las aplicaciones mediante unos servicios de transformación. Esto se hace en dos pasos, el adaptador convierte la información del formato de aplicación al formato común del bus. Y entonces se pueden aplicar transformaciones semánticas a esto (ejemplo: convirtiendo códigos postales a nombres de ciudades, separando/fusionando objetos de una aplicación en objetos de otras aplicaciones, y así sucesivamente).
- Módulos de integración: un sistema EAI puede participar en operaciones de integración concurrentes en un momento dado, cada tipo de integración es procesada por un módulo de integración diferente. Los módulos de integración se suscriben a eventos de tipos específicos y ellos reciben las notificaciones de procesos en el momento en que esos eventos ocurren.
- Soporte a transacciones: cuando se emplean para integración de procesos, el sistema EAI provee consistencia transaccional entre las aplicaciones al

ejecutar todas las operaciones que involucran una sola transacción distribuida (usando el protocolo de commit de dos fases o transacciones de compensación (operaciones que deshacen las acciones sobre un sistema dado). ***(Consultar Referencia URL[12])***

2.5.8 ARQUITECTURAS DE COMUNICACIÓN

Actualmente, hay muchas variaciones de pensamiento sobre lo que constituye la mejor infraestructura, modelo de componentes, y estándares de estructura para la Integración de aplicaciones empresariales. Parece haber consenso que cuatro componentes son esenciales para la arquitectura de una integración de aplicaciones empresarial:

1. Un broker centralizado que se encarga de la seguridad, el acceso y la comunicación. Esto se puede lograr a través de servidores de integración (como el Framework SIF) o por medio de programas similares, como el bus de servicios empresariales (ESB), modelo que actúa como un gestor de servicios orientados a SOAP.

2. Un modelo de datos independiente, basado en una estructura de datos estándar. Parece que XML y el uso de hojas de estilo XML se han convertido en estándares de facto.

3. Un conector, o modelo de agente donde cada proveedor, aplicación o interfaz pueda construir un solo componente que pueda hablar de forma nativa a esa aplicación y comunicarse con el broker centralizado.

4. Un modelo de sistema que define las API, flujo de datos y reglas de contrato con el sistema de tal manera que los componentes se pueden construir para interactuar con él de una manera estandarizada.

Aunque otros enfoques como la conexión a la base de datos o nivel de interfaz de usuario han sido explorados, no se han encontrado a la escala o ser capaces de adaptarse. Las aplicaciones individuales pueden publicar mensajes en el bróker centralizado y suscribirse para recibir mensajes de ese bróker. Cada aplicación requiere solamente una conexión con el bróker. Este enfoque central de control puede ser extremadamente escalable y altamente evolutivo.

La Integración de Aplicaciones Empresariales está relacionada con las tecnologías de middleware, tal como Middleware Orientado a Mensajes (MOM), y tecnologías de representación de datos, tal como XML. Otras tecnologías EAI involucran el uso de servicios web como parte de una arquitectura orientada a servicios, como medio de integración. La Integración de aplicaciones tiende a ser centralizado en los datos. En un futuro próximo, llegará a incluir la integración de contenidos y procesos de negocio.

2.5.9. PROBLEMAS DE IMPLEMENTACIÓN DE LOS EAI

En el año 2003 se reportó que el 70% de todos los proyectos EAI fallaron. La mayoría de dichas fallas no se debían a fallas técnicas del software mismo o la implementación, sino a problemas de gobernabilidad. El gerente general de EAIIC, Steve Craggs ha determinado los siete principales retos que afrontan las compañías que usan sistemas EAI y explica soluciones a dichos problemas.

- *Cambio constante*
  La propia naturaleza de EAI es dinámica y requiere dinámicos directores de proyecto para su aplicación.
- *Falta de expertise en EAI*
  EAI requiere conocimiento de muchas problemáticas y aspectos técnicos.
- *Estándares en competencia*
  Dentro del campo de EAI, la paradoja es que los estándares de EAI no son por si mismos universales, ya que cada proveedor particular trata de imponer los propios.
- *EAI es un paradigma de herramientas*
  EAI no es una herramienta, si no es un sistema y debe ser implementado como tal.

- *Construir interfaces es un arte*
  Realizar el proceso de ingeniería de la solución puede no ser suficiente. Las soluciones requieren ser negociadas con departamentos de usuarios para lograr un consenso común sobre el entregable final. La falta de consenso en los diseños de las interfaces tiende a acarrear un esfuerzo excesivo para mapear los requerimientos de datos de varios sistemas.
- Falta de detalle

La información que al principio parece poco importante, con el tiempo se puede volver crucial.

- Contabilidad
  Puesto que varios departamentos pueden tener requerimientos contradictorios entre sí, ellos deben contar para la estructura del sistema final.

Otros problemas potenciales pueden abarcar las siguientes áreas:

- *Requerimientos nuevos*
  Las implementaciones de EAI deben ser extensibles y modulares para permitir cambios futuros.
- *Proteccionismo*
  Las aplicaciones cuyos datos son integrados, frecuentemente pertenecen a departamentos diferentes los cuales tienen razones técnicas, culturales y políticas para no querer compartir su información con otros departamentos.

2.5.10 VENTAJAS Y DESVENTAJAS

- Ventajas:
  - Acceso a la información en tiempo real entre los sistemas.
  - Permite encadenar los procesos de negocio y ayuda a incrementar la eficiencia organizacional.
  - Mantiene la integridad de la información entre varios sistemas.
- Desventajas:
  - Costos de desarrollo muy altos, especialmente para pequeñas y medianas empresas (PYMEs)
  - Las implementaciones de EAI consumen mucho tiempo y requieren muchos recursos.
  - Requieren una gran cantidad de diseño frontal, el cual muchos gerentes son incapaces de visualizar o en el cual no muchos desean invertir. La mayoría de los proyectos de EAI usualmente comienzan como esfuerzos de integración punto a punto, y muy rápidamente se vuelven inmanejables en la medida que el número de aplicaciones aumenta.

2.5.11. EL FUTURO DE EAI

Las tecnologías de EAI aun están en desarrollo y no hay un consenso sobre cuál es el enfoque ideal o el grupo correcto de tecnologías que la compañía debería usar. ***(Consultar Referencia URL[12])***

2.6. WIRELESS APPLICATION PROTOCOL (WAP)

2.6.1 DEFINICIÓN

WAP (Wireless Application Protocol) es un protocolo basado en los estándares de Internet que ha sido desarrollado para permitir a teléfonos celulares navegar a través de Internet. Con la tecnología WAP se pretende que desde cualquier teléfono celular WAP se pueda acceder a la información que hay en Internet así como realizar operaciones de comercio electrónico.

WAP es una serie de tecnologías que consisten en: WML, WMLScript un lenguaje de script, lo que vendría a ser JavaScript y el Wireless Telephony Application Interface (WTAI).

2.6.2 TECNOLOGÍA WML

Lenguaje de etiquetas.
Las características principales de WML son:

- Soporte para imágenes y texto, con posibilidad de texto con formato.
- Tarjetas agrupadas en barajas. Una página WML es como una página HTML en la que hay una serie de cartas, al conjunto de estas cartas se les suele llamar baraja.
- Posibilidad de navegar entre cartas y barajas de la misma forma que se navega entre páginas Web.
- Manejo de variables y formularios para el intercambio de información entre el teléfono celular y el servidor.

Para los próximos años se calcula que el número de usuarios de WAP en el mundo crecerá , en parte, este crecimiento viene impulsado por la introducción del sistema General Packet Radio Service (GPRS), WAP 2.0, Bluetooth y el comercio móvil.

2.6.3 SISTEMA GPRS

Con el sistema GPRS (General Packet Radio Service) se puede disfrutar de una conexión inalámbrica continua a redes de datos y acceder a los servicios preferidos de información y entretenimiento. Sólo se tiene que conectar, sintonizar y descargar. GPRS es un excelente portador para muchos tipos de aplicaciones, como mensajería multimedia, imágenes y navegación.
GPRS, aumenta la velocidad de conexión y posibilita estar siempre conectado, no factura por tiempo de conexión sino por volumen de datos descargados.

Tecnología Usada por GPRS

GPRS utiliza la tecnología de conmutación por paquetes, en la que la información se transmite en pequeñas ráfagas de datos a través de una red basada en IP. GPRS es más adecuado para aplicaciones con transmisión de datos de carácter esporádico, EJ. Aplicaciones como servicios WAP, SMS, MMS y acceso a Internet. GPRS proporciona un rápido establecimiento de sesión y rapidez en la transmisión de datos. La seguridad de IP depende del operador y existen diversos métodos estándar de seguridad de Internet disponibles.

Debido a que los canales de comunicación para GPRS se utilizan a medida que se necesitan los paquetes (en lugar de dedicarse a un solo usuario cada vez), los usuarios deben localizar el servicio basado en paquetes GPRS de menor coste, aunque la facturación por GPRS depende del operador. Otra ventaja es que la mayor velocidad de transmisión de datos implica que no es necesario adaptarse a la menor velocidad de otros sistemas inalámbricos, por lo que se mejora la disponibilidad de aplicaciones para usuarios móviles, permitiéndose la conexión de los usuarios a ordenadores portátiles, interacción con sitios Web multimedia y aplicaciones similares.

2.6.4 POSIBILIDADES DE LA TECNOLOGÍA WAP

WAP ofrece infinitas posibilidades de cara tanto a empresas y profesionales como al consumidor:

Mediante el WAP podemos hacer lo siguiente:

- Comunicaciones Personales: E-mail, fax, SMS (servicio de mensajes cortos), Postales electrónicas, Mensajes Multimedia, Videotelefonía, Pizarra electrónica, etc.
- Oficina Móvil: Acceso a Internet e Intranets. Acceso a bases de datos corporativas. Videoconferencia FTP (transferencia de ficheros).
- Servicios de Información: Páginas Amarillas, Información del tráfico, Información turística (hoteles, agencias de viajes, paraderos,...), Horarios de trenes, aviones, etc. Mapas, Servicios de localización, Información de tiempo, tráfico.
- Servicios Personales: Gestión y consultas bancarias, Ticketing (compra de entradas para espectáculos), comercio electrónico móvil, Acceso a juegos y apuestas (loterías, quinielas, etc.), Carga de tarjetas de crédito (dinero electrónico), venta y reserva de billetes (transportes)

2.6.5 PLATAFORMA DE LA TECNOLOGÍA WAP

La plataforma WAP, se encarga de ofrecer un acceso inalámbrico seguro que ofrece acceso a un conjunto de servicios de Internet y a otras redes, a los usuarios que accedan mediante su teléfono móvil.

La utilización de un teléfono WAP es igual a la de un navegador Web: El usuario teclea para solicitar una URL. Pero, al contrario que los navegadores estándar que usan HTML para visualizar la información en la pantalla del ordenador, los teléfonos WAP utilizan WML, un lenguaje abierto desarrollado por el WAP Forum, que permite adaptarse a pequeños dispositivos de mano. Al igual que el HTML, WML se construye por medio de "tags" y permite la presentación de texto e imágenes, entrada de información y formularios.

El teléfono WAP utiliza las capacidades de información de conexiones inalámbricas convencionales para que el usuario realice peticiones al Gateway WAP. EL Gateway WAP convierte éstas en peticiones HTTP y las envía a través de Internet. Cuando el servicio requerido responde, el gateway WAP vuelve a enviar la información al teléfono WAP.

El Gateway WAP es el núcleo de la plataforma WAP. Su capacidad para actuar en esta clase de teléfonos como un Proxy HTTP, permite a los suscriptores acceder a cualquier site WWW. Algunos proveedores de información ofrecen igualmente servicios WML que usan WML para aprovechar la interfaz del teléfono WAP. Estos servicios pueden además iniciar la comunicación "impulsando" la información al gateway WAP, que como respuesta, transmite la misma a un teléfono WAP. Este proceso se denomina notificación.

Además de la translación HTML, la oferta de servicios del gateway varía. Estos pueden ser un servicio de protección de información por medio del mantenimiento de una base de datos de teléfonos WAP y sus privilegios de acceso, un servicio de fax que permitiese a los usuarios de teléfonos WAP mandar por fax contenido de un site Web a una máquina de fax local, o servicios de correo, organizadores o directorios. Todos ellos dependen de la suite de servicios que ofrezca cada Gateway.

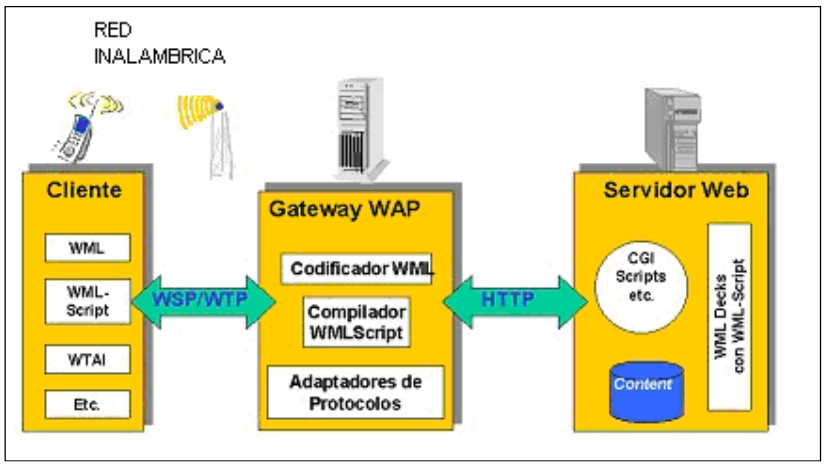


Figura 2.12: Plataforma de la Tecnología WAP

El Gateway WAP se encarga de dos labores:

1ª) Traducir la petición WAP, escrita en WML, a una petición WWW, permitiendo así que el cliente WAP pueda realizar peticiones al servidor Web.

2ª) Codificar las respuestas del servidor a un formato binario de modo que sea entendible por el cliente WAP.

El Servidor Web, que es el que se comunica con el Gateway WAP de dos posibles modos:

1º) Si el servidor Web proporciona un contenido WAP, como por ejemplo WML o WMLS, entonces no se necesita ningún filtro HTML.
2º) Si el servidor Web proporciona un servicio WWW, como HTML, entonces se usa un filtro HTML para traducir el contenido WWW en uno WAP. Como se ve en el ejemplo, el filtro HTML, puede encargarse de traducir una respuesta HTML en una WML, y devolvérsela al Gateway WAP.

2.6.6 FUNCIONAMIENTO DE LA TECNOLOGÍA WAP

La arquitectura de la plataforma WAP, está influida por la infraestructura y diseño de la "www" por lo que las transacciones WAP utilizan el mismo modelo básico, siendo la principal diferencia que el teléfono y el Gateway WAP sustituyen en conjunto al navegador Web.
WAP funciona de la siguiente manera:

1. El usuario solicita la página WAP que quiera ver.
2. El micronavegador del móvil envía la petición con la dirección (URL) de la página solicitada y la información sobre el abonado al Gateway WAP (software capaz de conectarse a la red de telefonía móvil y a Internet)
3. El Gateway WAP examina la petición y la envía al servidor donde se encuentra la información solicitada.
4. El servidor añade la información http o HTTPS pertinente y envía la información de vuelta al Gateway.
5. En el Gateway se examina la respuesta del servidor, se valida el código WML en busca de errores y se genera la respuesta que se envía al móvil.
6. El micronavegador examina la información recibida y si el código es correcto lo muestra en pantalla.

2.6.7 ARQUITECTURA DE LA TECNOLOGÍA WAP

La arquitectura WAP está pensada para proporcionar un "entorno escalable y extensible para el desarrollo de aplicaciones para dispositivos de comunicación móvil". Para ello, se define una estructura en capas, en la cual cada capa es accesible por la capa superior así como por otros servicios y aplicaciones a través de un conjunto de interfaces muy bien definidos y especificados.
Las capas de la arquitectura WAP se recogen en el siguiente diagrama:

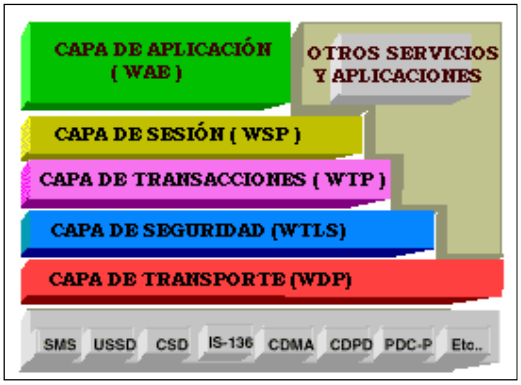


Figura 2.13: Arquitectura de la Tecnología WAP

A continuación se pasará a describir cada una de las capas:

CAPA DE APLICACIÓN (WAE):
Es un entorno de aplicación de propósito general, basado en la combinación del World Wide Web y tecnologías de Comunicaciones Móviles.
Este entorno incluye un micro navegador, que posee las siguientes funcionalidades:
El lenguaje WML, antes mencionado.
El lenguaje WMLS, similar al JavaScript.

WTA (Wireless Telephony Applicactions), es un entorno para aplicaciones u servicios de telefonía.

WTAI (Wireless Telephony Application Interface), es una interfaz utilizada en los terminales móviles para operaciones locales de control de llamadas (recepción, iniciación y terminación) y acceso a listines telefónicos.

Una serie de formatos de contenido, que son un conjunto de datos definidos, entre los que se encuentran: imágenes, información de calendario.

CAPA DE SESIÓN (WSP)

Este protocolo proporciona a la Capa de Aplicación (WAE) interfaz con dos servicios de sesión:

- Un servicio orientado a conexión que funciona por encima de la Capa de Transacciones (WTP).
- Un servicio no orientado a conexión que funciona por encima de la Capa de Transporte (WTP), y que proporciona servicio de datagramas seguro o no seguro.

Esta capa proporciona las siguientes funcionalidades:

- Establecimiento y liberación de conexiones entre cliente y servidor.,
- Intercambio de información entre cliente y servidor.
- Negociación de las características del protocolo.
- Suspensión y reanudación de la sesión.

CAPA DE TRANSACCIONES (WTP)

Este protocolo funciona por encima de un servicio de datagramas ya sean seguros como no seguros, y proporciona las siguientes funcionalidades:

Proporciona los servicios necesarios para soportar las transacciones, estos servicios pueden ser de tres clases:

- Peticiones inseguras de un solo camino
- Peticiones seguras de un solo camino
- Transacciones seguras de dos caminos

Asimismo proporciona seguridad en las transacciones.

CAPA DE SEGURIDAD (WTLS)

La Capa Inalámbrica de Seguridad de transporte (WTLS) es un protocolo basado en el estándar SSL, utilizado en el entorno Web, para la seguridad en la transferencia de datos, esta capa proporciona a las capas de nivel superior de Wap una interfaz de servicio de transporte seguro, que lo resguarde de una interfaz de transporte inferior.

Las funcionalidades de esta capa son las siguientes:

- **Integridad de los datos**: se asegura que la información intercambiada entre el Terminal y el servidor de aplicaciones, no haya sido modificada.
- **Privacidad de los datos**: se asegura que la información intercambiada entre el Terminal y el servidor de aplicaciones, no pueda ser captada ni entendida por elementos externos a la comunicación.
- **Autentificación**: se ofrecen servicios para determinar la autenticidad del Terminal y del servidor de aplicaciones.
- También puede ser utilizado para el establecimiento de una comunicación segura entre terminales.

CAPA DE TRANSPORTE (WDP)

El Protocolo Inalámbrico de Datagramas (WDP) proporciona las siguientes funcionalidades:

- Proporciona un servicio fiable a los protocolos de las capas superiores de WAP.
- Permite la comunicación de forma transparente sobre los protocolos portadores: CDMA, SMS, GSM...

2.7. ENTERPRISE SERVICE BUS (ESB)

Un Enterprise Service Bus es una infraestructura de software que funciona como una capa intermedia (middleware), proporcionando servicios de integración de las distintas aplicaciones a través de mensajería basada en estándares y servicios de sincronización. ESB ocupa la capa de abstracción intermedia entre los distintos sistemas de una o varias organizaciones, proporcionando mecanismos de comunicación y transformación de mensajes basados en estándares.

ESB debe ser capaz de reemplazar todo el contacto directo entre las aplicaciones, consiguiendo que todas ellas se comuniquen a través del bus.

Los ESB transmiten y reciben mensajes basados en estándares, pero deben ser capaces de transformar mensajes a formatos que sean reconocidos por las distintas

aplicaciones en el caso que sea necesario, lo que se realiza a través de adaptadores. Además, el intercambio de mensajes debe ser independiente de la plataforma. Esto permite al ESB integrar aplicaciones que se ejecuten en diversos sistemas operativos o mainframes.

ESB trata a todas las aplicaciones como servicios, con independencia de cómo se conecten al bus, permitiendo a las empresas migrar paulatinamente a una arquitectura basada en servicios con un riesgo mínimo y una eficaz planificación de las inversiones

Una arquitectura orientada a los servicios (SOA) es un método para construir una infraestructura de TI a partir de componentes acoplados de manera flexible, denominados "servicios", que desempeñan funciones específicas. Las aplicaciones compuestas son un elemento clave de un entorno SOA. Estas aplicaciones se crean invocando y orquestando múltiples servicios, eventos y modelos de tal manera que colectivamente desempeñan una función empresarial de alto nivel. El principal problema es cómo resolver la escalabilidad de las conexiones punto a punto, lo que se conoce también como el "problema de conexión M*N". Si tenemos múltiples aplicaciones orientadas a servicio, a la hora de diseñar como consumir los servicios unas de otras (siendo el número de combinaciones punto a punto exponencial), el problema se simplifica si todas las aplicaciones se conectan a un punto único y central de conexión: el bus de servicios empresariales.

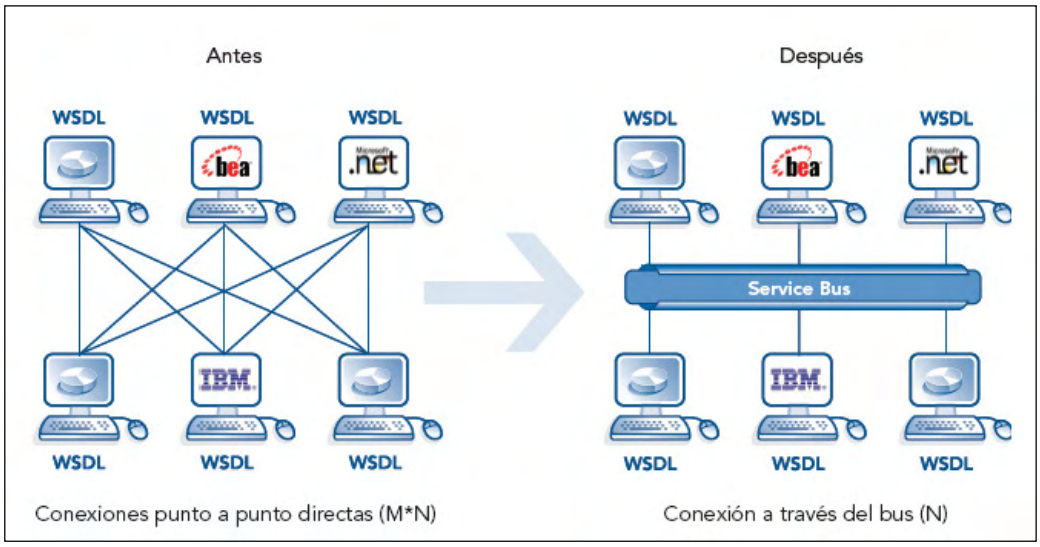


Figura 2.14: Conexión Punto a punto Vs Servicio Bus

Características

- Independientes respecto a sistemas operativos y lenguajes de programación
- Uso general del XML como lenguaje estándar de comunicación
- Soporte de estándares de Servicios Web
- Adaptadores para realizar la integración con aplicaciones
- Transformación de mensajes
- Validación de mensajes
- Enrutamiento de mensajes aplicando reglas de negocio y en función de contenido del mensaje.
- Manipulación de excepciones
- Soporte a encolado y mantenimiento de mensajes, si las aplicaciones no están disponibles
- Seguridad e integración entre las aplicaciones

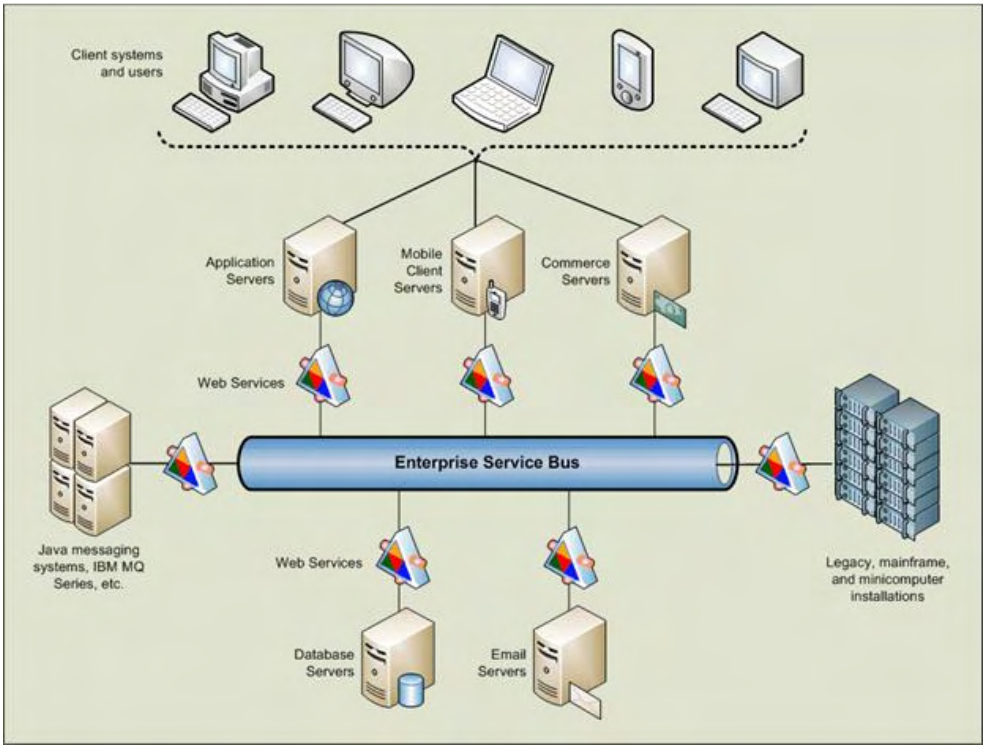


Figura 2.15: Arquitectura ESB

Las soluciones ESB incluyen más que SOA y WebService, son orquestadores de servicios (pero en el sentido amplio de la palabra, no solo webService), permiten interactuar de mejor forma y más rápidamente a las componentes técnicas y de información con aquellas relacionadas a los procesos de negocio de las capas.

El aporte de un ESB va mas allá de una herramienta que facilite integraciones del tipo SOAP/HTTP WebServices sobre redes TCP/IP, habilita middleware de servicios orientados a mensajería como JMS, administra colas y prioriza datas y comunicaciones de contenidos, administrando los estados de estas integraciones entre sistemas distribuidos. ESB aplica y apoya servicios débilmente acoplados, esquemas de eventos y listens, acceso a repositorios sobre múltiples plataformas y repositorios de información. ***(Consultar Referencia URL[35])***

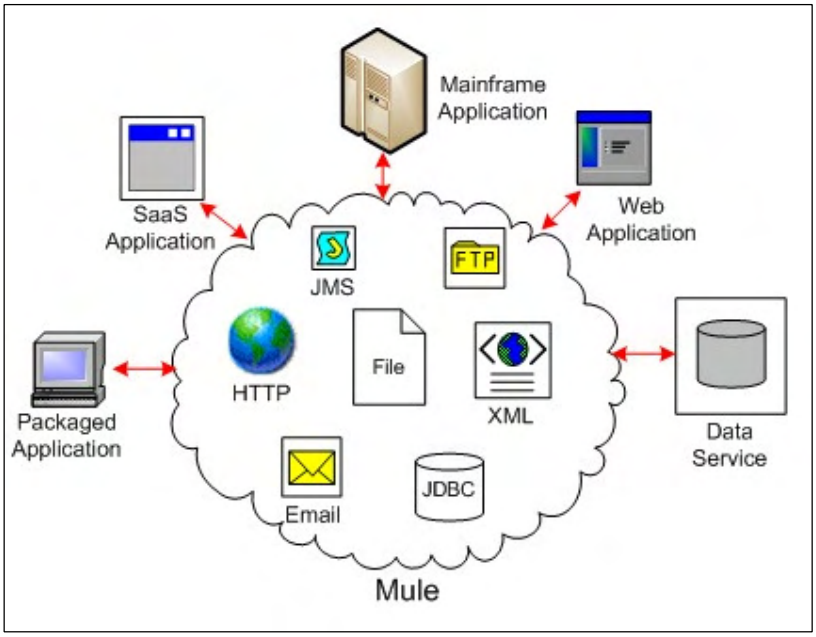


Figura 2.16: Comunicación ESB

2.8. JAVA 2 MICRO EDITION (J2ME)

2.8.1. INTRODUCCIÓN

Al principio de los 90, Sun Microsystems creó un nuevo lenguaje de programación llamado Oak como parte de un proyecto de investigación para construir productos electrónicos que dependan principalmente del software. El primer prototipo para Oak fue un controlador portable llamado Star7, un pequeño dispositivo handheld con una pantalla touchscreen LCD que tenía incorporado soporte a redes inalámbricas y comunicaciones infrarojas. Este dispositivo podría ser usado como control remoto para televisores o VCR y como guía de programas electrónicos, e incluso tenía algunas funciones que ahora son asociadas a los PDAs, como agenda

de citas. El software para este tipo de dispositivos necesitaba ser extremadamente confiable y no debía hacer excesivo uso de memoria ni requerir demasiada potencia en el procesador. Oak fue desarrollado como resultado de la experiencia del equipo de desarrollo con el lenguaje C++, el cual, a pesar de tener muchas grandes características, demostró que era un lenguaje complejo y ocasionaba que los programadores cometan fácilmente errores y eso afectaba la confiabilidad del software.

Oak fue diseñado para quitar o reducir la posibilidad de que los programadores cometan errores, ¿cómo? detectando la mayoría de errores en tiempo de compilación y quitando algunas de las características del lenguaje C++ (como punteros y la administración de memoria controlada por el programador) que eran los problemas más comunes.

Desafortunadamente, el mercado para el tipo de dispositivos que el nuevo lenguaje fue creado no se desarrolló tanto como Sun Microsystems esperaba, y al final ningún dispositivo basado en Oak fue vendido a los clientes. Sin embargo, al mismo tiempo, el inicio del conocimiento público de Internet produjo un mercado para el software de navegación para Internet (los navegadores Web). En respuesta a esto, Sun Microsystems renombró el lenguaje de programación Oak a Java y lo usó para desarrollar un navegador multiplataforma llamado HotJava. También le dio la licencia de Java a Netscape, quienes lo incorporaron en su navegador que por ese entonces era el más popular en el mercado, luego fueron incorporados los Java Applets

.

En un par de años, las capacidades multiplataforma del lenguaje de programación Java y su potencia como plataforma de desarrollo para aplicaciones que podían ser escritas una vez y ejecutadas en diversos sistemas Windows y Unix, había despertado el interés de usuarios finales, porque vieron en ella una manera de reducir los costos del desarrollo de software.

2.8.2. DEFINICIÓN

En la conferencia JavaOne de 1999 Sun Microsystem reconoció la premisa que dice que “con un único tamaño no es posible abarcarlo todo” y decidió reagrupar toda la tecnología Java en torno a varias ediciones para abarcar desde el rango desde las vastas aplicaciones distribuidas a lo largo y ancho de la red hasta el diminuto mundo de las aplicaciones ejecutables en dispositivos móviles. Partiendo pues de esta decisión, Sun definió el lenguaje Java simplemente como Plataforma Java 2 y ha redistribuido esta plataforma en tres ramas en función del sector al que va orientada la edición correspondiente, cada rama con su conjunto de APIs y herramientas de desarrollo propias. Sun Microsystem distingue:

Java2 Standard Edition (J2SE), orientada a ordenadores de sobremesa. Comprende el JDK hasta ahora distribuido por Sun, en donde Swing se ha convertido en pieza clave y al que se han incorporado clases adicionales para facilitar el desarrollo de aplicaciones Java en donde la interfaz de usuario tiene una importancia muy especial. Esta es la versión en la que la mayoría de la gente piensa cuando piensa en Java.

Java 2 Enterprise Edition (J2EE), engloba al J2SE y lo potencia añadiéndole clases para el desarrollo en entornos corporativos. Esta edición de la plataformaJava 2 está orientada al desarrollo de aplicaciones para servidores utilizandp Enterprise Java Beans, aplicaciones web, Servlets, Java Server Pages, CORBA y Extensible Markup Language (XML). Es decir, esta edición está más orientada al desarrollo de componentes y distribución de aplicaciones, luciendo toda la parafernalia asociada a las aplicaciones al nivel de negocio. ***(Consultar Referencia Libros - [2])***

Java 2 Micro Edition (J2ME), Es un subconjunto de J2SE orientado al desarrollo de aplicaciones Java destinadas a dispositivos con pocos recursos, con capacidades restringidas, tanto con respecto a la capacidad de memoria disponible, limitaciones de la pantalla gráfica como con respecto a la capacidad de procesamiento; características típicas de cualquier equipo de electrónica de consumo por ejemplo, teléfonos celulares, PDAs, buscapersonas, mensáfonos, dispositivos de navegación de coches, etc.

Sun Microsystems respondió a esta demanda creando varias plataformas Java con funcionalidades reducidas, cada una hecha a la medida de un segmento vertical y específico del mercado.

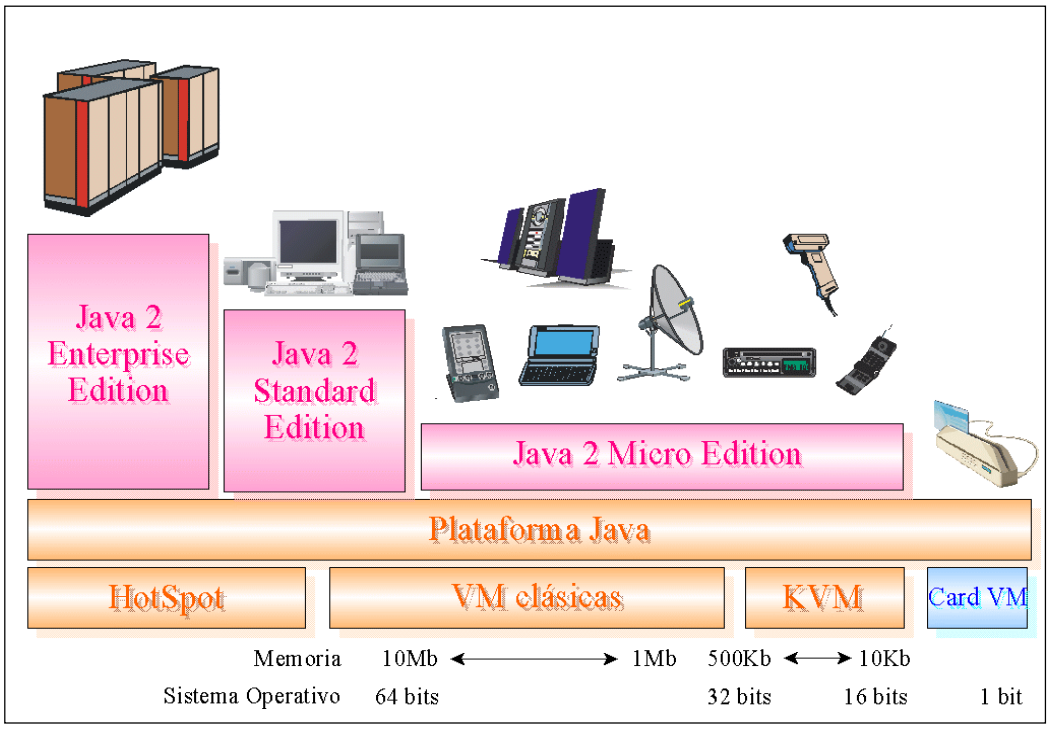


Figura 2.17: Arquitectura de la plataforma Java 2 de Sun

KVM Corresponde con la Máquina Virtual más pequeña desarrollada por Sun. Su nombre KVM proviene de Kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40Kb y 80Kb). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. La KVM está escrita en lenguaje C, aproximadamente unas 24000 líneas de código

2.8.3. CONFIGURACIONES

Conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Consiste en un entorno de ejecución Java completo que define el entorno de ejecución básico de J2ME, su objetivo es adecuarse a las necesidades de una familia de dispositivos con capacidades similares. Una configuración es una

especificación, define una clase de dispositivos en términos de hardware: tipo y velocidad del procesador, capacidad de memoria, tipo de conectividad en redes Describen las características básicas, comunes a todos los dispositivos

- Características soportadas del lenguaje de programación Java.
- Características soportadas por la Máquina Virtual Java.
- Bibliotecas básicas de Java y APIs soportadas.

.Existen dos configuraciones en J2ME: CDC y CLDC

Configuración de dispositivos con conexión, CDC (Connected Device Configuration)

Está orientada a dispositivos con cierta capacidad computacional y de memoria. Por ejemplo, decodificadores de televisión digital, televisores con internet, algunos electrodomésticos y sistemas de navegación en automóviles. CDC usa una Máquina Virtual Java (CVM Compact Virtual Machine) similar en sus características a una de J2SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo. La CDC está enfocada a dispositivos con las siguientes capacidades:

- Procesador de 32 bits.
- Disponer de 2 Mb o más de memoria total, incluyendo memoria RAM y ROM.
- Poseer la funcionalidad completa de la Máquina Virtual Java2.
- Conectividad a algún tipo de red.

La CDC está basada en J2SE v1.3 e incluye varios paquetes Java de la edición estándar. Las peculiaridades de la CDC están contenidas principalmente en el paquete javax.microedition.io, que incluye soporte para comunicaciones http y basadas en datagramas.

• Configuración de dispositivos limitados con conexión, CLDC (Connected Limited Device Configuration).

La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Un ejemplo de estos dispositivos

son: teléfonos móviles, buscapersonas (pagers), PDAs, organizadores personales, etc.

Está orientado a dispositivos con ciertas restricciones, algunas de estas restricciones vienen dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño. Los dispositivos que usan CLDC deben cumplir los siguientes requisitos:

- Disponer entre 160 Kb y 512 Kb de memoria total disponible. Como mínimo se debe disponer de 128 Kb de memoria no volátil para la Máquina Virtual Java y las bibliotecas CLDC, y 32 Kb de memoria volátil para la Máquina Virtual en tiempo de ejecución.
- Procesador de 16 o 32 bits con al menos 25 Mhz de velocidad.
- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.
- Tener conexión a algún tipo de red, normalmente sin cable, con conexión
- Intermitente y ancho de banda limitado (unos 9600 bps).

La CLDC aporta las siguientes funcionalidades a los dispositivos:

- Un subconjunto del lenguaje Java y todas las restricciones de su Máquina
- Virtual (KVM).
- Un subconjunto de las bibliotecas Java del núcleo.
- Soporte para E/S básica.
- Soporte para acceso a redes.
- Seguridad.

2.8.4. PERFILES

Define las APIs que controlan el ciclo de vida de la aplicación, interfaz de usuario, etc. Más concretamente, un perfil es un conjunto de APIs orientado a un ámbito de aplicación determinado. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan (electrodomésticos, teléfonos móviles, etc.) y el tipo de aplicaciones que se ejecutarán en ellos. Las librerías de la interfaz gráfica son un componente muy importante en la definición de un perfil. Aquí nos podemos encontrar grandes diferencias entre interfaces, desde el menú textual de los teléfonos móviles hasta los táctiles de los PDAs. El perfil establece unas APIs que definen las características de un dispositivo, mientras que la configuración hace lo propio con una familia de ellos. Esto hace que a la hora de construir una aplicación

se cuente tanto con las APIs del perfil como de la configuración. Tenemos que tener en cuenta que un perfil siempre se construye sobre una configuración determinada. De este modo, podemos pensar en un perfil como un conjunto de APIs que dotan a una configuración de funcionalidad específica

El principal perfil y el más utilizado por haber sido el primero del cual se ha proporcionado una implementación es el perfil MIDP, basado en la configuración CLDC para ejecutar aplicaciones en teléfonos celulares, buscapersonas, dispositivos con pantallas reducidas, conexión HTTP inalámbrica y memoria limitada.

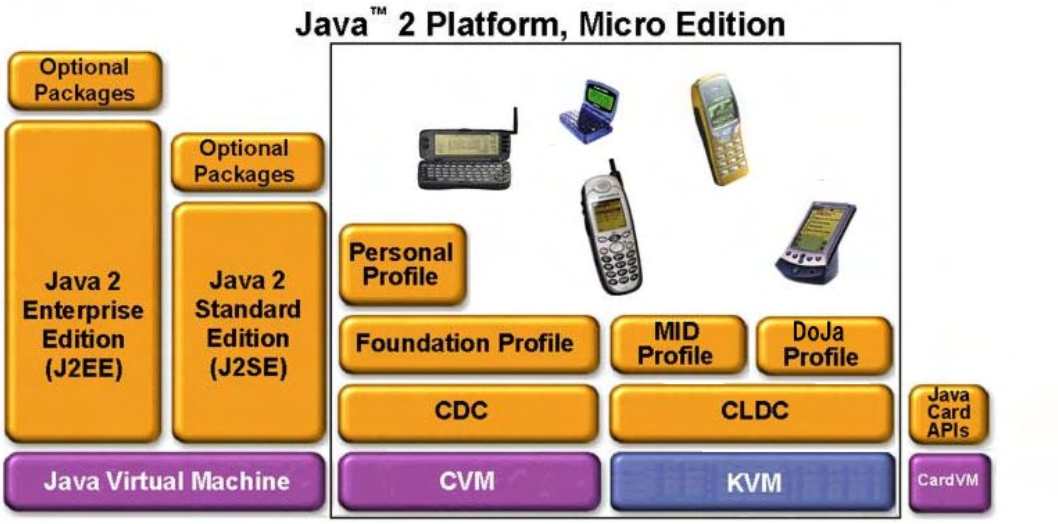


Figura 2.18: Arquitectura de Capas J2ME

2.8.5. MIDLET

Son aplicaciones creadas usando las especificaciones CLDC y MIDP. Están diseñados para ser ejecutados, en dispositivos con poca capacidad gráfica, de cómputo y de memoria

2.9. MODELO DE LAS 4+1 VISTAS

La Arquitectura Software trata el diseño e implementación de la estructura de alto nivel del software. Es el resultado de ensamblar un cierto número de elementos arquitectónicos para satisfacer la funcionalidad y ejecución de los requisitos del sistema; así como los requisitos no funcionales del mismo: fiabilidad, escalabilidad, portabilidad, disponibilidad, etc. Perry y Wolf (1992) describen una arquitectura software como:

Arquitectura Software = {Elementos, Formas, Fundamento/Restricciones}

Es muy complejo capturar la arquitectura software en un sólo modelo (o diagrama). Para manejar esta complejidad se representan diferentes aspectos y características de la arquitectura en múltiples vistas. Una vista es "una presentación de un modelo, la cual es una descripción completa de un sistema desde una particular perspectiva" . El modelo más aceptado a la hora de establecer las vistas necesarias para describir una arquitectura software es el modelo 4+1 (Kruchten, 1995***. Consultar Referencia URL [5]***). Este modelo define 4 vistas principales:

- Vista Lógica (Logical View), modelo de objetos, clases, entidad – relación, etc.
- Vista de Procesos (Process View), modelo de concurrencia y sincronización.
- Vista de Implementación (Implementation View), organización estática del software en su entorno de desarrollo (librerías, componentes, .ear, .jar, etc.).
- Vista Física o Despliegue (Deployment View), modelo de correspondencia software, hardware (aspectos de distribución en máquinas, por ejm)
- Vista Caso de Uso (Use - Case View, Vista "+1"). Muestra y traza en cada una de las anteriores y que está formada por las necesidades funcionales que cubre el sistema, y que en ocasiones identificamos como vista de "casos de uso". De donde deducimos que según este modelo, la arquitectura es en realidad evolucionada desde escenarios.

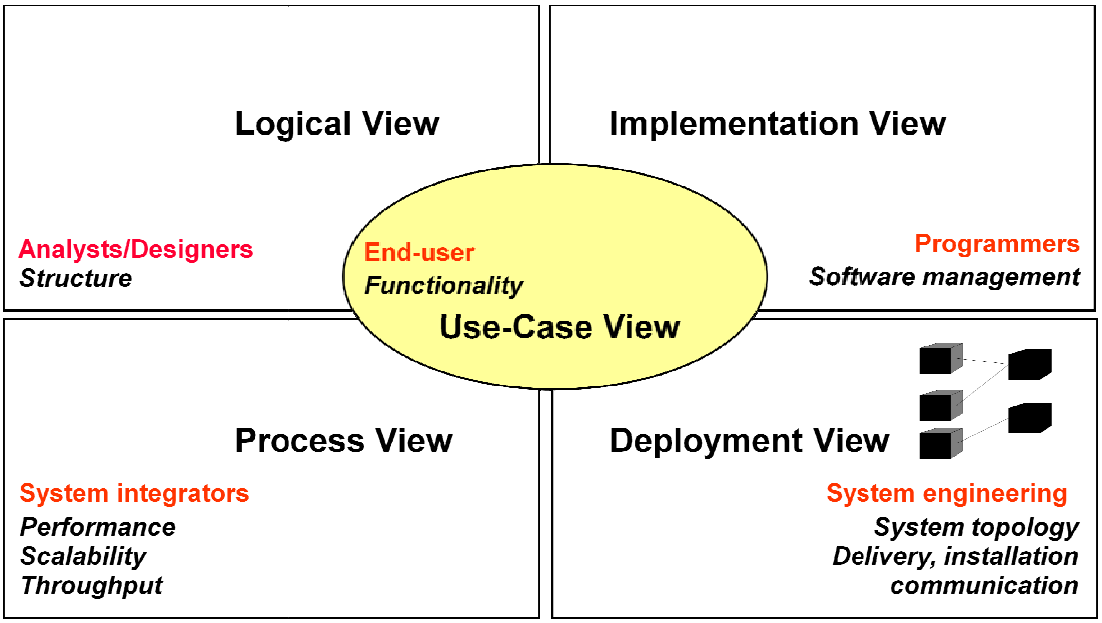


Figura 2.19: Modelo 4+1 Vistas

CAPÍTULO III

3. PLANTEAMIENTO DE LA SOLUCIÓN

3.1 ANTECEDENTES DE LA SOLUCIÓN

Los primeros antecedentes de la implementación de un Sistema integrador de Bibliotecas pertenecen a la Oficina Nacional de Bibliotecas Públicas (ONBP), que funcionaba en la sede de la Biblioteca Nacional del Perú y administraba el Fondo San Martín.

El año de 1968 se puso en funcionamiento el Catálogo Unido de la ONBP, llamado "Listado IBM", y que contenía aproximadamente 6000 registros utilizando cintas perforadas que era emitida por una máquina flaxowriter de la época.

Para el año 1971 con la misma infraestructura se obtiene el "Listado de Epígrafe" con 5330 registros con envíos incluyentes y excluyentes; el año de 1979 un grupo de alumnos de la Escuela Nacional de Bibliotecarios dirigidos por Marta Bryce, delinea los primeros pasos para la planificación del Sistema Mecanizado de información para Bibliotecas Públicas.

El año de 1980 las Bibliotecarias Ada Rengifo y Antonieta Arecco, compilan un Sistema terminológico para constituir el archivo temático del Sistema con aproximadamente 10000 registros y sus respectivos envíos. Para ese mismo año se estaba trabajando con registros maestros de cerca de 35000 fichas bibliográficas.

En el año 2005 este proyecto cobra vigorosidad y se plantean una serie de actividades:

- Posicionamiento del SNB (Sistema Nacional de Bibliotecas) para la obtención de compromisos políticos con las autoridades municipales de Lima y Provincias.
- Decisión política para la implementación de los Centros Coordinadores y proyecto del Catálogo Unido Automatizado del SNB (Declaratoria de Emergencia del SNB)
- Identificación de las bibliotecas que cuentan con registros bibliográficos (diversos soportes) para su integración en el catálogo.

- Capacitación de los responsables de las bibliotecas municipales para la tarea de marcado y selección de registros.
- Acopio de los registros de las bibliotecas municipales seleccionadas para el Catálogo Unido Automatizado del SNB.
- Presentación del catálogo en línea en la página web de la Biblioteca.

3.2 PROPUESTAS DE SOLUCIONES ACTUALES

3.2.1 SOA Vs EAI (INTEGRACIÓN DE APLICACIONES)

- SOA: Es un concepto de arquitectura de software que define la utilización de servicios para facilitar la integración entre diferentes aplicaciones independientes del lenguaje y la plataforma.
- EAI: Se define como el uso de software y principios de arquitectura de sistemas para integrar un conjunto de aplicaciones.

Comparativa

EAI:

- Costos de desarrollo muy altos, especialmente para pequeñas y medianas empresas
- Las implementaciones de EAI consumen mucho tiempo y requieren muchos recursos.
- Requieren una gran cantidad de diseño frontal, el cual muchos gerentes son incapaces de visualizar o en el cual no muchos desean invertir. La mayoría de los proyectos de EAI usualmente comienzan como esfuerzos de integración punto a punto, y muy rápidamente se vuelven inmanejables en la medida que el número de aplicaciones aumenta.

Ventajas:

- Acceso a la información en tiempo real entre los sistemas.
- Permite encadenar los procesos de negocio y ayuda a incrementar la eficiencia organizacional.
- Mantiene la integridad de la información entre varios sistemas.

SOA:

- Se requiere un alto esfuerzo.

- La velocidad de intercambio de información entre sistemas es más lenta que una conexión directa, intercambiar grandes volúmenes de información puede afectar al rendimiento del bus.

Ventajas:

- Mejora en los tiempos de realización de cambios en procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.
- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (socios, proveedores).
- Poder para reemplazar elementos de la capa aplicativa SOA sin disrupción en el proceso de negocio
- Escalabilidad
- Robustez
- Facilidad en la adaptación de nuevos servicios
- Facilidad en la reestructuración de sistemas

EAI envuelve varios adaptadores entre componentes que son dependientes de la tecnología, por eso es complejo y costoso de mantener. Los adaptadores del programa están generalmente basados sobre APIs y formatos de archivos, pero estos cambian, conduciendo a la inestabilidad del sistema.

En **SOA**, cada componente utiliza la misma manera para comunicarse con otros componentes, basado sobre estándares independientes de la plataforma. Este enfoque simplifica y consolida la integración eficientemente

La siguiente tabla muestra una comparación de las arquitecturas SOA y EAI:

Parámetro de Evaluación	**EAI Arquitectura HUB**	**EAI Arquitectura BUS**	**SOA Arquitectura ESB**
Esfuerzo de Instalación	Menos esfuerzo de instalación comparado con la arquitectura en Bus	Esfuerzo moderado	Esfuerzo moderado
Administración	Fácil de mantener y administrar debido al hub central	La administración puede ser compleja dependiendo de los sistemas integrados	La administración puede ser compleja dependiendo de los sistemas

			integrados
Costo	Alto	Alto	Bajo Costo debido a que no utiliza formatos propietarios para mejorar el rendimiento. Aunque no provee todos los servicios usualmente proporcionados por las suites de productos propietarias
Escalabilidad	Alta si se utiliza una arquitectura federada, de otra manera es limitada por el hardware utilizado para alojar un Hub	Altamente escalable	Altamente escalable
Estándares	En su mayoría basado en estándares pero puede usar formatos propietarios internos	En su mayoría basado en estándares pero puede usar formatos propietarios internos	Basado en estándares
Orientado a Servicios	Puede ser implementado como orientado a servicios	Puede ser implementado como orientado a servicios	Orientado a servicios

Conclusión

Para la integración d aplicaciones utilizaremos la arquitectura SOA por los siguientes motivos:

- Bajo costo
- Basado en estándares
- No dependencia de tecnologías propietarias
- Orientación a servicios

- Utilización de web services los cuales son independientes de la plataforma, mientras que en EAI no hay independencia
- Bajo costo de desarrollo en comparación a la EAI
  (Consultar Referencia URL[13])

3.2.2 WEBSERVICES Vs REST

WEB SERVICES

El consorcio W3C define los Servicios Web como sistemas software diseñados para soportar una interacción interoperable maquina a máquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja.

La definición de Servicios Web propuesta alberga muchos tipos diferentes de sistemas, pero el caso común de uso de refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar SOAP.
La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Estándares empleados

- Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- XML (Extensible Markup Language): Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call): Protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- WSDL (Web Services Description Language): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- UDDI (Universal Description, Discovery and Integration): Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.

- WS-Security (Web Service Security): Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.

REST

La Transferencia de Estado Representacional (Representational State Transfer) o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP, y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. Aunque REST no es un estándar, está basado en estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG/…
- Tipos MIME: text/xml, text/html,…

Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de un interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

Una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida, no necesitando conocer si existen cachés, proxys,

cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. La aplicación, sin embargo, debe comprender el formato de la información devuelta (la representación), que es por lo general un documento HTML o XML, aunque también puede ser una imagen o cualquier otro contenido.
Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST. ***(Consultar Referencia URL [32])***

Comparativa

REST es un estilo de arquitectura que basicamente explota la tecnología existente y los protocolos de la Web, incluyendo HTTP (Hypertext Transfer Protocol) y XML. REST es más fácil de usar que SOAP (Simple Object Access Protocol), sin embargo SOAP ofrece potencialmente más capacidad.

REST es un estilo, mientras que los servicios Web son sistemas software. Por tanto, no es posible la comparación de ambos conceptos. Por otra parte, popularmente se generaliza el concepto de servicio Web con el de servicio Web basado en SOAP. Como hemos visto en apartados anteriores, es posible diseñar servicios Web basados en REST, es decir tomando REST como estilo de diseño. Por lo tanto se compara los web services basadas en SOAP y los web services basados en REST o RestFul Web Services. ***(Consultar Referencia URL [30])***

WEBSERVICES RESTful

En los últimos años se ha popularizado un estilo de arquitectura Software conocido como REST (Representational State Transfer). Este nuevo estilo ha supuesto una nueva opción de estilo de uso de los Servicios Web.

Los Servicios Web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar (por ejemplo GET, PUT,…). Por tanto, este estilo se centra más en interactuar con recursos con estado, que con mensajes y operaciones. ***(Consultar Referencia URL [29])***

Cómo diseñar un servicio Web basado en REST

Realizar los siguientes pasos:

- Identificar todas las entidades conceptuales que se desean exponer como servicio.

- Crear una URL para cada recurso. Los recursos deberían ser nombres no verbos (acciones). Por ejemplo no utilizar esto:

  http://www.service.com/entities/getEntity?id=001

  Como podemos observar, getEntity es un verbo. Mejor utilizar el estilo REST, un nombre: http://www.service.com/entities/001

- Categorizar los recursos de acuerdo con si los clientes pueden obtener un representación del recurso o si pueden modificarlo. Para el primero, debemos hacer los recursos accesibles utilizando un HTTP GET. Para el último, debemos hacer los recursos accesibles mediante HTTP POST, PUT y/o DELETE.

- Todos los recursos accesibles mediante GET no deberían tener efectos secundarios. Es decir, los recursos deberían devolver la representación del recurso. Por tanto, invocar al recurso no debería ser el resultado de modificarlo.

- Ninguna representación debería estar aislada. Es decir, es recomendable poner hipervínculos dentro de la representación de un recurso para permitir a los clientes obtener más información.

- Especificar el formato de los datos de respuesta mediante un esquema (DTD, W3C Schema, ect). Para los servicios que requieran un POST o un PUT es aconsejable también proporcionar un esquema para especificar el formato de la respuesta.

- Describir como nuestro servicio ha de ser invocado, mediante un documento WSDL/WADL o simplemente HTML. ***(Consultar Referencia URL[30])***

Comparativa

Según lo que hemos descrito, el principal beneficio de SOAP recae en ser fuertemente acoplado, lo que permite poder ser testeado y depurado antes de poner en marcha la aplicación. En cambio, las ventajas de la aproximación basada en REST recaen en la potencial escalabilidad de este tipo de sistemas, así como el acceso con escaso consumo de recursos a sus operaciones debido al limitado número de operaciones y el esquema de direccionamiento unificado.

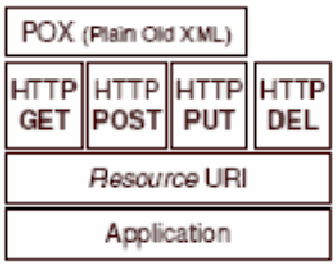
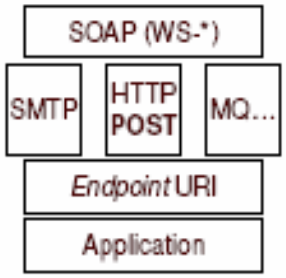
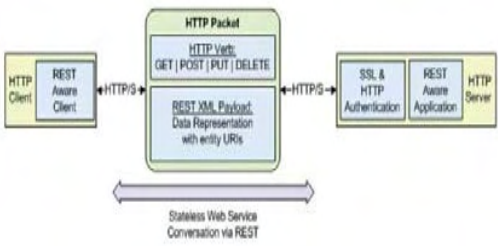
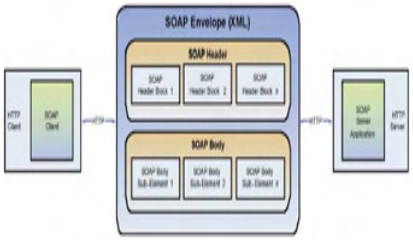
A modo de resumen, veamos las características de ambas aproximaciones en la siguiente tabla:

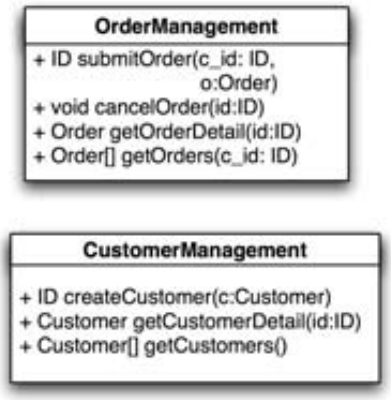
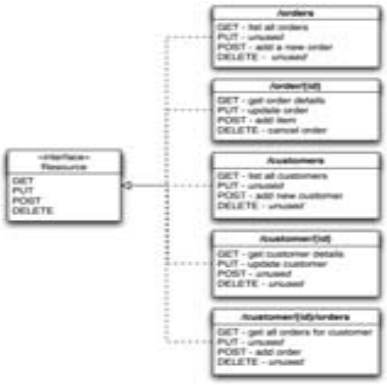
	REST	SOAP
Características	Las operaciones se definen en los mensajes. Una dirección única para cada instancia del proceso. Cada objeto soporta las operaciones estándares definidas. Componentes débilmente acoplados.	Las operaciones son definidas como puertos WSDL. Dirección única para todas las operaciones. Múltiple instancias del proceso comparten la misma operación. Componentes fuertemente acoplados.
Ventajas declaradas	Bajo consumo de recursos. Las instancias del proceso son creadas explícitamente. El cliente no necesita información de enrutamiento a partir de la URI inicial. Los clientes pueden tener una interfaz "listener" (escuchadora) genérica para las notificaciones. Generalmente fácil de construir y adoptar.	Fácil (generalmente) de utilizar. La depuración es posible. Las operaciones complejas pueden ser escondidas detrás de una fachada. Envolver APIs existentes es sencillo Incrementa la privacidad. Herramientas de desarrollo.
Posibles desventajas	Gran número de objetos. Manejar el espacio de nombres (URIs) puede ser engorroso. La descripción sintáctica/semántica muy informal (orientada al usuario). Pocas herramientas de desarrollo.	Los clientes necesitan saber las operaciones y su semántica antes del uso. Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. Las instancias del proceso son creadas implícitamente.

La principal diferencia entre REST y SOAP se resume en los siguientes puntos de vista del propósito de la Web:

REST	SOAP
“La Web es el universo de la información accesible globalmente” (Tim Berners Lee)	“La Web es el transporte universal de mensajes”

A continuación vamos a intentar esbozar las diferencias entre REST y SOAP desde varios puntos de vista:

	RE	SOAP
Tecnología	Interacción dirigida por el usuario por medio de	Flujo de eventos orquestados.
	Pocas operaciones con muchos recursos	Muchas operaciones con pocos recursos.
	Mecanismo consistente de nombrado de recursos (URI).	Falta de un mecanismo de nombrado.
	Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	Se centra en el diseño de aplicaciones distribuidas.
Protocolo		
		
	XML autodescriptivo.	Tipado fuerte, XML Schema.
	HTTP.	Independiente del transporte.
	HTTP es un protocolo de aplicación.	HTTP es un protocolo de transporte.
	Síncrono.	Síncrono y Asíncrono.

Descripción del servicio	Confía en documentos orientados al usuario que define las direcciones de petición y las respuestas.	WSDL.
	Interactuar con el servicio supone horas de testeado y depuración de URIs.	Se pueden construir automáticamente stubs (clientes) por medio del WSDL.
	No es necesario el tipado fuerte, si ambos lados están de acuerdo con el contenido.	Tipado fuerte.
	WADL propuesto en noviembre de 2006.	WSDL 2.0.
Gestión del estado	El servidor no tiene estado (stateless).	El servidor puede mantener el estado de la conversación.
	Los recursos contienen datos y enlaces representando transiciones a estados válidos.	Los mensajes solo contienen datos.
	Los clientes mantienen el estado siguiendo los enlaces.	Los clientes mantienen el estado suponiendo el estado del servicio.
	Técnicas para añadir sesiones: Cookies	Técnicas para añadir sesiones: Cabecera de sesión (no estándar)
Seguridad	HTTPS.	WS-Security.
	Implementado desde hace muchos años.	Las implementaciones están comenzando a aparecer ahora.
	Comunicación punto a punto segura.	Comunicación origen a destino
Metodología de diseño		
	Identificar recursos a ser expuestos como servicios.	Listar las operaciones del servicio en el documento WSDL.
	Definir URLs para direccionarlos.	Definir un modelo de datos para el contenido de los mensajes.
	Distinguir los recursos de solo lectura (GET) de los modificables (POST, PUT, DELETE).	Elegir un protocolo de transporte apropiado y definir las correspondientes políticas QoS, de seguridad y transaccional.

	Implementar e implantar el servidor Web.	Implementar e implantar el contenedor del servicio Web.

Dónde es útil REST?

Tanto los arquitectos como los desarrolladores necesitan decidir cuál es el estilo adecuado para las aplicaciones. En algunos casos es adecuado un diseño basado en REST, se listan a continuación:

- El servicio Web no tiene estado. Una buena comprobación de esto consistiría en considerar si la interacción puede sobrevivir a un reinicio del servidor.
- Una infraestructura de caching puede mejorar el rendimiento. Si los datos que el servicio Web devuelve no son dinámicamente generados y pueden ser cacheados, entonces la infraestructura de caching que los servidores Web y los intermediarios proporcionan, pueden incrementar el rendimiento.
- Tanto el productor como el consumidor del servicio conocen el contexto y contenido que va a ser comunicado. Ya que REST no posee todavía (aunque hayamos visto una propuesta interesante) un modo estándar y formal de describir la interfaz de los servicios Web, ambas partes deben estar de acuerdo en el modo de intercambiar de información.
- El ancho de banda es importante y necesita ser limitado. REST es particularmente útil en dispositivos con escasos recursos como PDAs o teléfonos móviles, donde la sobrecarga de las cabeceras y capas adicionales de los elementos SOAP debe ser restringida.
- La distribución de Servicios Web o la agregación con sitios Web existentes puede ser fácilmente desarrollada mediante REST. Los desarrolladores pueden utilizar tecnologías como AJAX y toolkits como DWR (Direct Web Remoting) para consumir el servicio en sus aplicaciones Web. ***(Consultar Referencia URL [33])***

¿Dónde es útil SOAP?

Un diseño basado en SOAP es adecuado cuando:

- Se establece un contrato formal para la descripción de la interfaz que el servicio ofrece. El lenguaje de Descripción de Servicios Web (WSDL), como ya sabemos, permite describir con detalles el servicio Web.

- La arquitectura debe abordar requerimientos complejos no funcionales. Muchas especificaciones de servicios Web abordan tales requisitos y establecen un vocabulario común para ellos. Algunos ejemplos incluyen transacciones, seguridad, direccionamiento,… La mayoría de aplicaciones del mundo real se comportan por encima de las operaciones CRUD y requieren mantener información contextual y el estado conversacional. Con la aproximación REST, abordar este tipo de arquitecturas resulta más complicado.

- La arquitectura necesita manejar procesado asíncrono e invocación. En estos casos, la infraestructura proporcionada por estándares como WSRM y APIs como JAX-WS junto con la asincronía por el lado del cliente nos permitirán el soporte de estas características.
  (Consultar Referencia URL [34])

Conclusión

Como arquitectura para web sevices, utilizaremos los web services basados en SOAP.

3.2.3 WAP Vs J2ME

El usuario podrá consultar mediante su dispositivo móvil los materiales bibliográficos de la biblioteca; para realizar el requerimiento es necesario escoger una de las dos tecnologías más fuertes que están presentes en el mercado WAP y J2ME.

Cada vez más empresas están desarrollando productos en tecnologías móviles: diarios, empresas de consumo masivo, versiones móviles de sitios web, banca móvil, etc. Podemos identificar que hay dos tendencias en este aspecto hacer un sitio WAP o desarrollar un cliente rico (por lo general en J2ME por su alta portabilidad). Ambos esquemas presentan sus ventajas y desventajas, los cuales analizaremos a continuación.

WAP

Figura 3.1: WAP La Nación

Ventajas

- **Alto grado de penetración.** La mayoría de los celulares tienen un navegador WAP con mayores o menores prestaciones, pero lo tienen. (Figura 3.1)
- **Complejidad de desarrollo moderado/bajo.** Hay muchas plataformas para desarrollar portales WAP, PHP o Visual Studio.Net son algunas de ellas.
- **Fácil distribución de actualizaciones.** Si necesitamos hacer un cambio en la aplicación simplemente debemos actualizar el sitio WAP.

Desventajas

- **Interfaces pobres.** Los sitios WAP son bastantes pobres, ya que por lo general la calidad gráfica no es buena, el ancho del sitio no concuerda con el ancho del navegador, son lentos, muchos no soportan Javascript o flash por lo que no es posible lograr buenos efectos.
- **No permiten navegación off-line.** Si uno ya entró a un sitio determinado y quiere volver a acceder al mismo sitio y no se cuenta con señal no se va a poder acceder al mismo nuevamente.
- **Tráfico de datos alto.** Este tipo de aplicaciones tienen una alta demanda de tráfico de datos ya que todo elemento que vemos tiene que ser descargado, además si ya bajamos información y la queremos ver de nuevo tenemos que descargarla nuevamente. Este es importante ya que el mecanismo de facturación de tráfico de datos es por Kb.
- **No tienen acceso al hardware del celular** Esto es una gran restricción ya que nos perdemos de usar Bluetooth, cámara de fotos, envío de SMS, GPS, gráficos en 3D, etc., lo que termina limitando el tipo de aplicaciones que podemos llegar a desarrollar.

J2ME

Figura 3.2: Gmail Aplicación J2ME

Ventajas

- **Rápida navegación.** Puesto que la lógica de la presentación (Interfaces de usuario) se encuentran en el mismo móvil, no son enviados desde el servidor, de esta manera se tiene al servidor sólo envía la información útil.
- **Mejor experiencia de usuario.** Se pueden hacer desarrollos que tengan animaciones, 3D, grafica avanzada, etc. Además tienen mayor tiempo de respuesta cuando se navega entre distintas pantallas, ya que no está todo el tiempo descargando datos de la web. (Figura 3.2)
- **Permiten navegación off-line.** La aplicación puede descargar la información de Internet y luego persistirla para poder leerla cuando queramos sin necesidad de estar conectados.
- **Tráfico de datos menores.** Pueden almacenar y procesar los datos a nivel local, reduciendo así el tráfico de red. Esto no sólo conserva el ancho de banda inalámbrico y reduce la latencia, se reduce la probabilidad de que la información crucial será interceptada o interrumpida (por ejemplo, por la negación de los ataques al servicio). El tráfico de ve reducido por varios factores, como por ejemplo el hecho de persistir información en el celular para su posterior relectura, podemos tener precargadas todas las imágenes,

los datos que descargamos de Internet es solamente información y no información con formato, etc.

- **Acceso al hardware** Posibilidad de tener acceso a todo el hardware del celular (Siempre y cuando las API necesarias estén soportadas), podemos hacer aplicaciones que usen Bluetooth, envíen SMSs, usen la cámara de fotos, etc.

Desventajas

- **Menor grado de penetración** Es menor la cantidad de dispositivos que soportan J2ME comparándolo con los que tienen un navegados WAP. Además el IPhone y el reciente Android no tienen soporte para esta plataforma, por lo que hay que pensar en tener versiones de nuestra aplicación para este tipo de dispositivos.
- **Un poco más complejo la distribución de actualizaciones** Al momento de desarrollar la aplicación tenemos que considerar algún mecanismo de actualización de la aplicación automático, en caso contrario la actualización sería manual, ingresando a una dirección WAP y descargarla.

Conclusiones

Para el sistema propuesto es mejor tener un cliente rico (Lógica de presentación en el dispositivo) J2ME puesto que la navegación es rápida y eficiente en el sentido de recibir del servidor sólo la información útil y necesaria. La experiencia en la parte gráfica que puede lograr el usuario se ve mucho mas enriquecida y las posibilidades de soluciones creativas son mucho mayores que en un sitio WAP.

Por la parte económica, el usuario se ve muy beneficiado ya que la facturación en el tráfico de datos es menor (Sin traer imágenes, textos, animaciones, etc.)

3.2.4 WEB SERVICE PUNTO A PUNTO Vs ESB

Los servicios de consulta de materiales de libros de cada biblioteca municipal serán publicados haciendo uso de WebServices, así cualquier biblioteca municipal podrá acceder a los servicios de otra biblioteca y viceversa.
El problema estriba en la forma de comunicación entre todos los servicios de las bibliotecas, por un lado tenemos a los accesos punto a punto entre cada sistema de biblioteca municipal y el otro es usando un middleware ESB

WEB SERVICE PUNTO A PUNTO

- Tiene una inversión inicial baja, pero el mantenimiento se hace costoso
- El número de conexiones crece exponencialmente a medida que se incorporan aplicaciones en nuevas municipalidades
- Requiere desarrollo de funciones complejas para comunicación, transformaciones, etc.
- Altamente dependiente de las aplicaciones, cada sistema debe tener presente la dirección ip y el nombre de todas los demás webservices de todas las otras municipalidades.

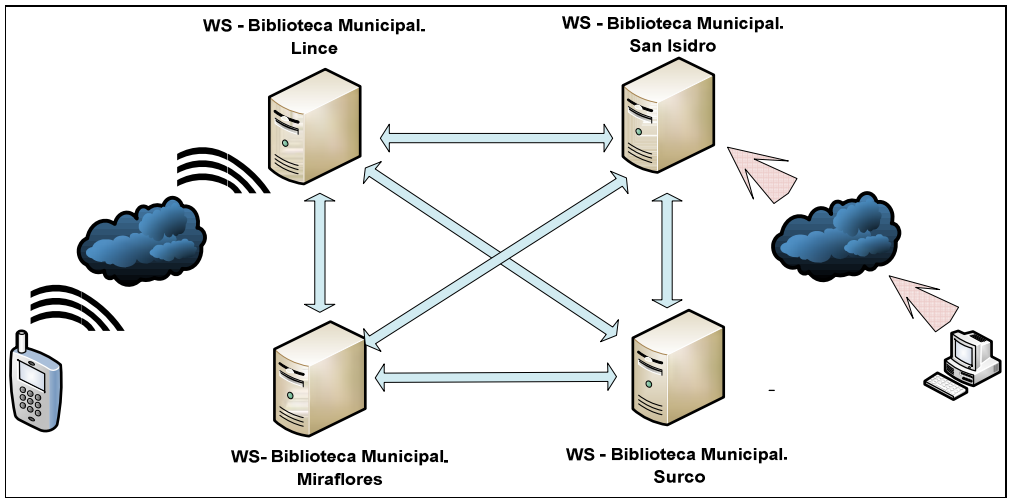


Figura 3.3: Comunicación WebService Punto a Punto

ENTERPRISE SERVICE BUS (ESB)

- Utiliza infraestructuras basadas en estándares
- La escalabilidad depende en gran medida de la infraestructura utilizada
- Flexibilidad en utilizar otras opciones de conectividad para incorporar cualquier tipo de sistema (JDBC, FTP, JMS, SMTP entre otros).
- Puede desarrollarse incrementalmente, ingresando más puntos de conexión, en nuestro caso de estudio los web services de las municipalidades
- Ofrece funciones de comunicación y transformación de datos , puesto que el ESB contiene transformadores para cuando exista comunicación entre 2 aplicaciones de diferente protocolo
- Integración a bajo costos, a diferencia de los EAI que son muy costosos el mantenimiento.
- A cada aplicación de una municipalidad le es indiferente la dirección ip y el nombre del webService de las otras municipalidades, sólo se comunica con el ESB y éste se encarga de realizar la conectividad.

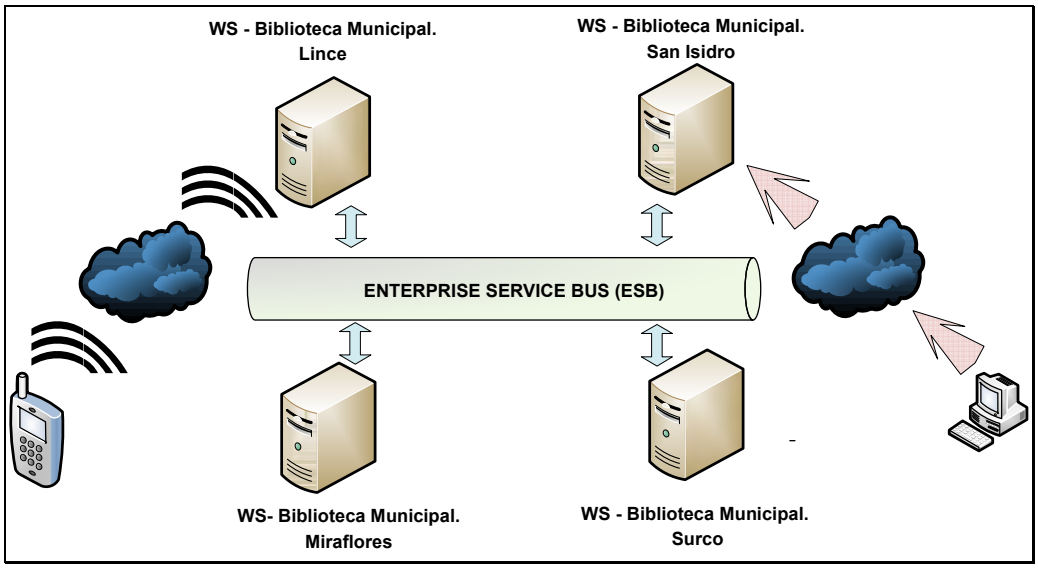


Figura 3.4: Comunicación WebService a través de un ESB

Conclusiones

- La arquitectura aplicada para la inter comunicación de servicios sería la del ESB puesto que existe un mayor orden en la comunicación, y la independencia del sistema de cada municipalidad en conocer la dirección y el nombre del web service de las otras.
- Además con esta infraestructura es una puerta abiera para cuando las municipalidades deseen intercambiar no sólo información mediante los webservices sino con otros protocolos como FTP, JMS, etc.

3.2.5 SELECCIÓN DEL FRAMEWORK ESB

Se consideró los siguientes productos entre Open Source y comerciales

PRODUCTO	VENDEDOR	CONECTIVIDAD
Matrix BusinessWorks	TIBCO	SOAP, EMS,JMS, Rendezvous,MQ, BPEL
Mule ESB	Open-source, MuleSource	SOAP, REST, JMS, MQ, JBI, AQ, Caching, JavaSpaces, GigaSpaces, Email, IM, JCA, AS400 Data Queues, System I/O.
OpenESB	Open-source, Sun Microsystems	JBI, JCA, JAX-RPC, JAX-WS
Sonic ESB	Progress Software	JMS, SOAP, JMX
Websphere ESB	IBM	JMS, MQ, SOAP; requiera adaptaroes adiciones a la interfaz, con otros productos o procolos legales; requiere el Websphere para trabajar.

Considerando factores como la tecnología existente, del desarrollo y del despliegue, política de la compañía, sociedades estratégicas (o carencia de ellas), se decidió por seleccionar el framework basado en los siguientes criterios:

- Comunidad activa de código abierto y soporte comercial disponible.
- Estado del arte de la implementación y habilidad que funciona sobre Java 5/6.

- Número de otras características relevantes fuera de las descritas en el cuadro.
- Facilidad de instalación y desarrollo.
- Facilidad de configuración y expansión.
- Capacidad para integrar dentro/fuera sin incurrir en el bloqueo o dependencias del producto dependientes.
- Coste total bajo por parte del propietario.

El framework Mule (http://www.mulesoft.org) es altamente escalable, lo que le permite comenzar conectando pocas aplicaciones y después conectar una vasta cantidad de aplicaciones en el tiempo. Administra todas las interacciones entre las aplicaciones y componentes de forma transparente, independientemente de si existen en la misma máquina virtual o en Internet, e independientemente del protocolo de transporte subyacente utilizado. En resumen las características principales del framework Mule son ***(Consultar Referencia URL[4]):***

- Sigue el patrón IoC (Inyección de dependencias)
- Ofrece una capa de mensajería, sincronía y asíncrona, así como adaptadores a las principales tecnologías de comunicaciones de datos (JMS, JDBC, TCP, UDP, multicast, http, servlet, SMTP, POP3, file, XMPP).
- Ofrece la posibilidad de modelado de servicios así como el registro y descubrimiento de los mismos.
- Dispone de adaptadores que facilitan la interacción con la capa superior de la pila SOA, la capa de orquestación de servicios y procesos.
- En cuanto a gestión de servicios, tiene adaptadores a contenedores JBI y a otras herramientas de gestión de servicios, como Xfire/CXF, Axis y Glue que facilitan el descubrimiento o registro de servicios.
- En cuanto a seguridad, soporta transacciones multiprotocolo y ofrece adaptadores para el control de acceso y la encriptación de información.
- Integración con servidor de aplicaciones JBoss
- Integración con Spring
- Adaptadores para disparar procesos de forma planificada y configurable: Quartz.
- Adaptadores para la interacción con Web Services.

3.3 ARQUITECTURA DEL SOFTWARE (MODELO DE LAS 4+1 VISTAS)

3.3.1 VISTA DE CASO DE USO

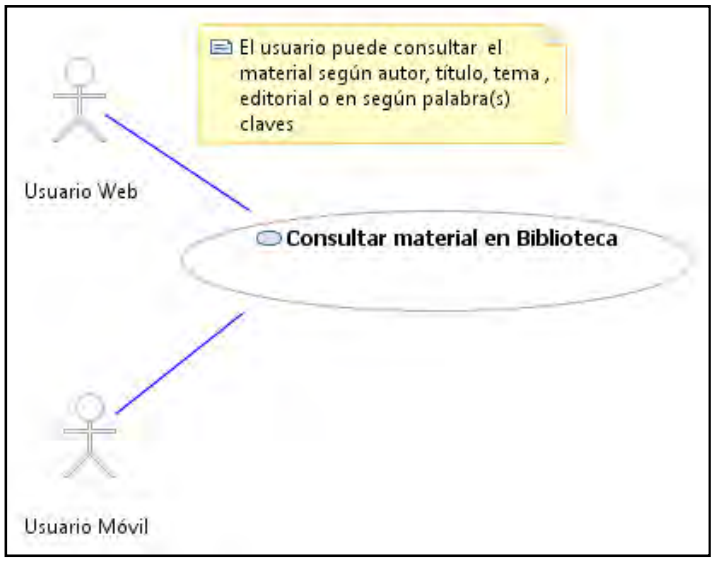


Figura 3.5: Vista Caso de Uso

TÉRMINOS	DEFINICIÓN	
Caso de uso	Consultar material en Biblitoteca	
Descripción general	Realiza la búsqueda de materiales bibliográficos en la biblioteca local	
Pre – condición	Ninguna	
Post – condición	Ninguno.	
Actores	Usuario Web, Usuario móvil	
Flujo Principal	1	El actor ingresa al sistema y selecciona la biblioteca en la cual quiere acceder, ingresa los criterios de búsqueda: por título, autor o tema , así como también si desea las opciones de búsqueda avanzada el cual contempla el uso de conecores lógicos.
	2	Si ha seleccionado la opción “TODOS” de la lista de selección, Se inicia el flujo alternativo “Buscar material en todas las bibliotecas.”

	3	El actor presiona el botón de búsqueda
	4	El sistema muestra un listado con todos los materiales encontrados de acuerdo al filtro seleccionado
	5	Si el actor desea consultar el detalle de cada material selecciona uno de la lista mostrada. Se inicial el flujo alternativo ”Consultar detalle material bibliográfico”
Flujo Alternativo Buscar material en todas las bibliotecas	1	El actor selecciona la opción “TODOS” de la lista de selección y presiona el botón de búsqueda
	2	El sistema muestra un listado con todos los materiales encontrados de acuerdo al filtro seleccionado, indicando en una columna a qué biblioteca pertenece el libro
	3	Si el actor desea consultar el detalle de cada material selecciona uno de la lista mostrada. Se inicial el flujo alternativo de uso “Consultar detalle material bibliográfico”
Flujo Alternativo Consultar Detalle Material bibliográfico	1	El usuario selecciona un registro de la lista de bibliotecas municipales y presiona el botón para ver el detalle
	2	El sistema muestra el detalle del libro con su información correspondiente.
Puntos de extensión		Ninguno.
Excepciones		**No existe conexión con el WebService de una municipalidad**
	1	El sistema muestra un mensaje de error informando que no existe conexión con el servidor
Relación con otros casos de uso	Ninguno	

3.3.2 VISTA LÓGICA

DIAGRAMA DE CLASES

Paquete WebService (Publicado en la cada Municipalidad)

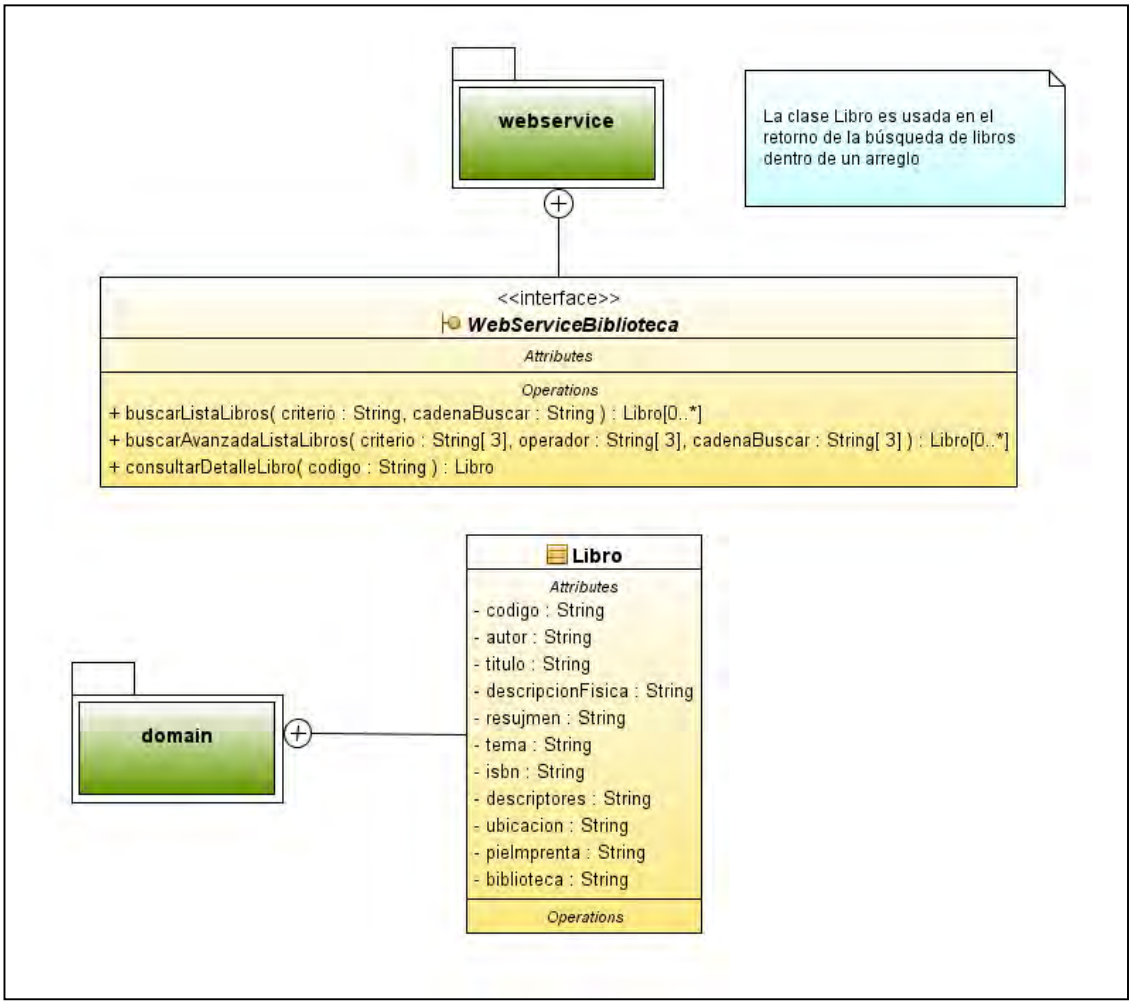


Figura 3.6: Diagrama del paquete WebService

En el sistema integrador estará implementado el cliente del WebService pero existirá un solo cliente para todas las municipalidades.

Paquete Service (En el Sistema Integrador y en cada Sistema Municipal)

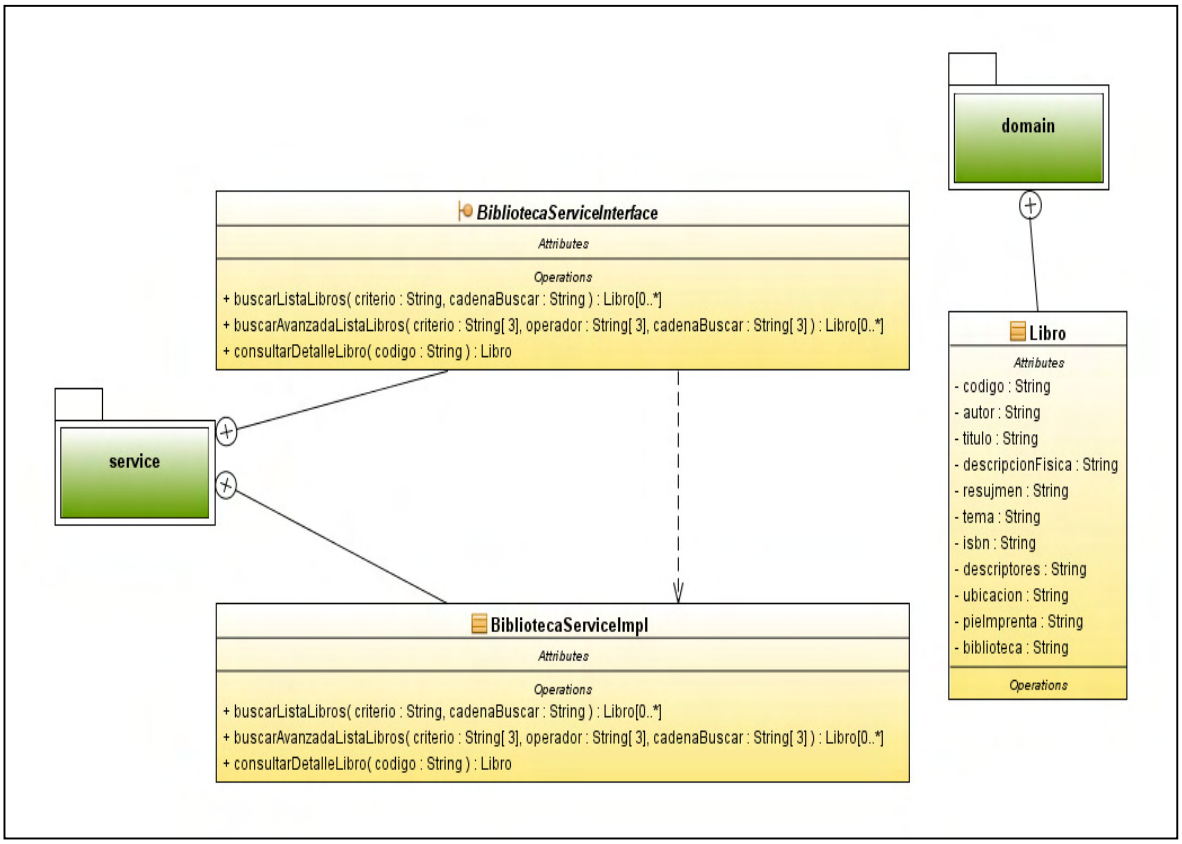


Figura 3.7: Diagrama del paquete Service

Este diagrama de clases también es el mismo que el que tendrá el Sistema Integrador de Bibliotecas.

No existe modelo de datos puesto que no tendremos una base de datos propia del sistema, en su lugar se utilizará la base de datos con la que cuenta el sistema de bibliotecas que cuenta el municipio

DIAGRAMA DE SECUENCIA

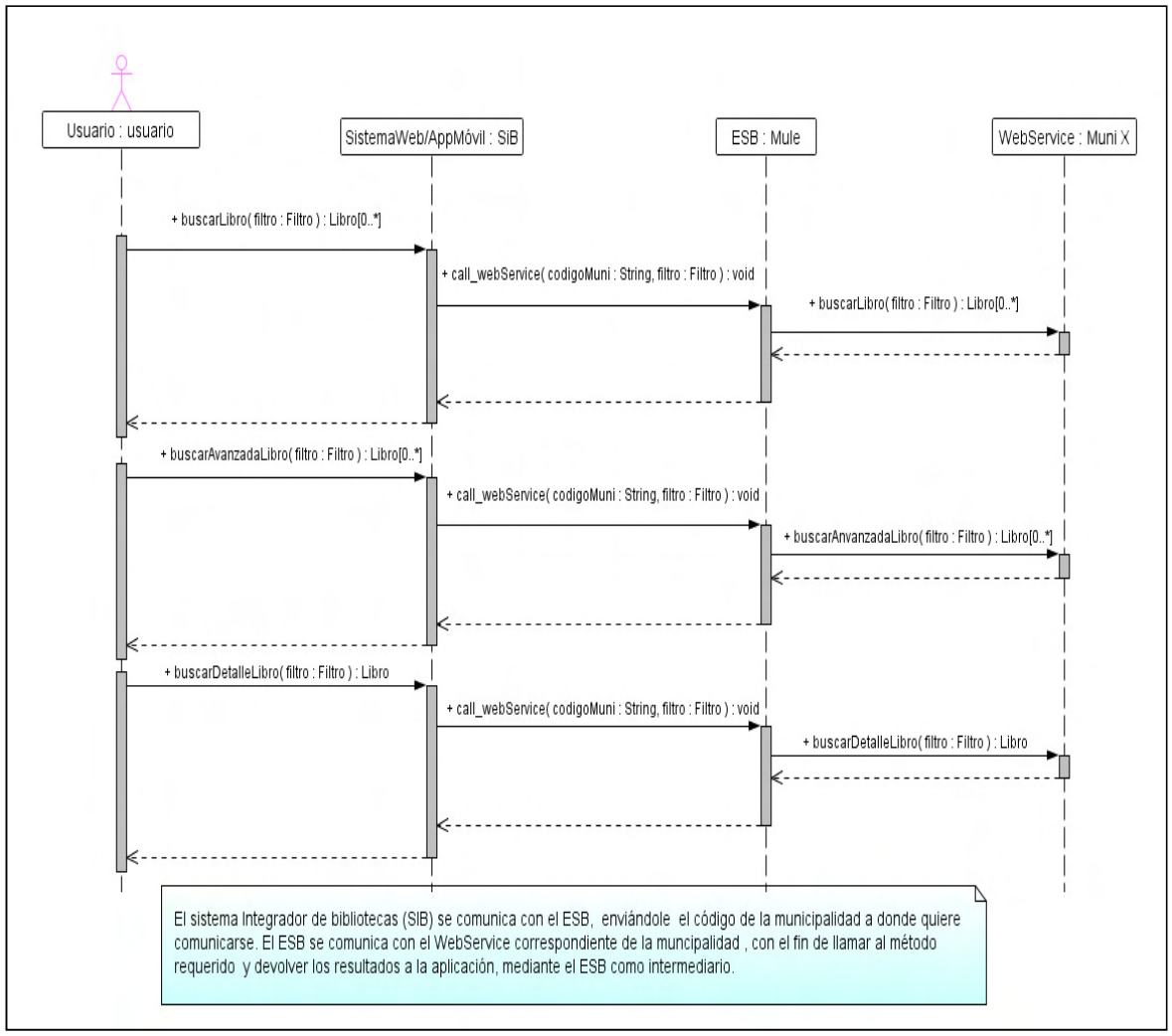


Figura 3.8: Diagrama de Secuencia

3.3.3 VISTA DE IMPLEMENTACIÓN

DIAGRAMA DE COMPONENTES

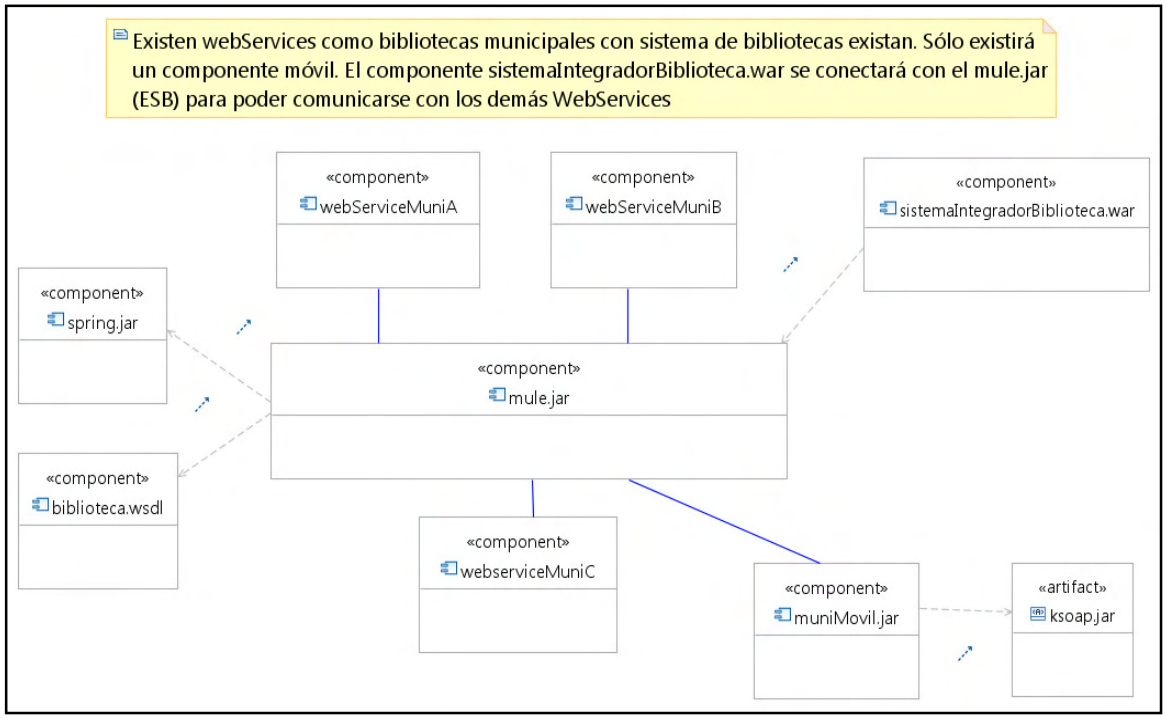


Figura 3.9: Diagrama de Componentes

3.3.4 VISTA FÍSICA

DIAGRAMA DE DESPLIEGUE

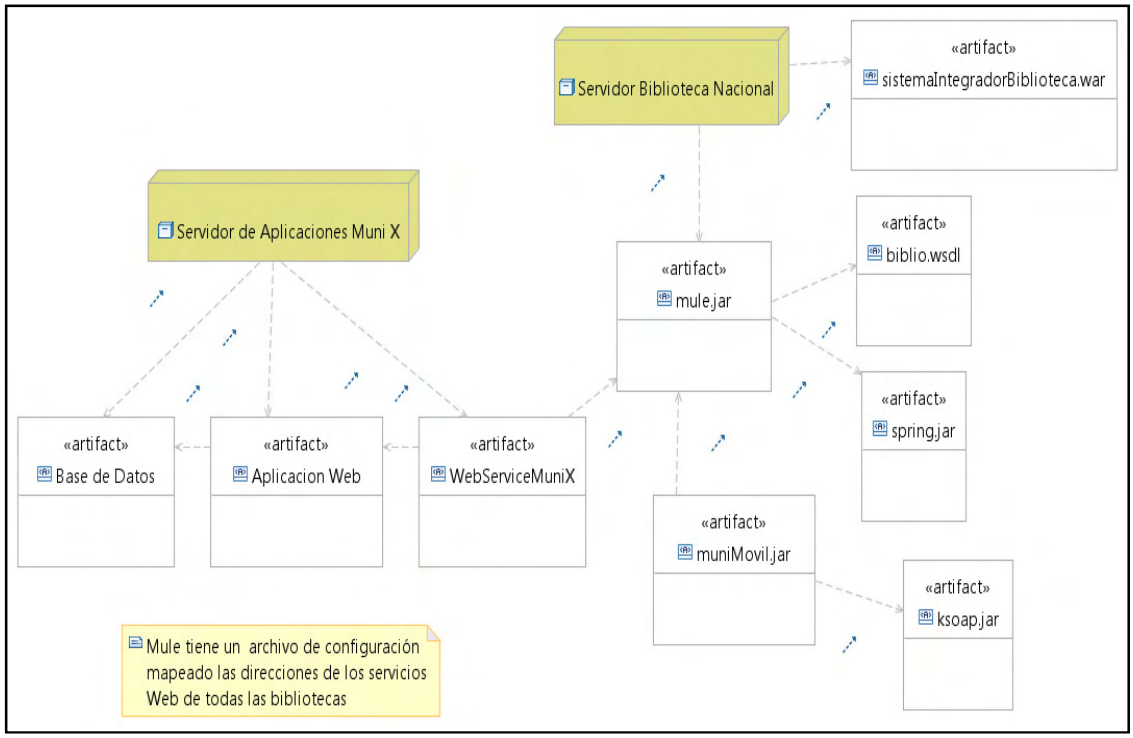


Figura 3.10: Diagrama de Despliegue

3.3.5 VISTA DE PROCESOS

DIAGRAMA DE PROCESOS A NIVEL MACRO

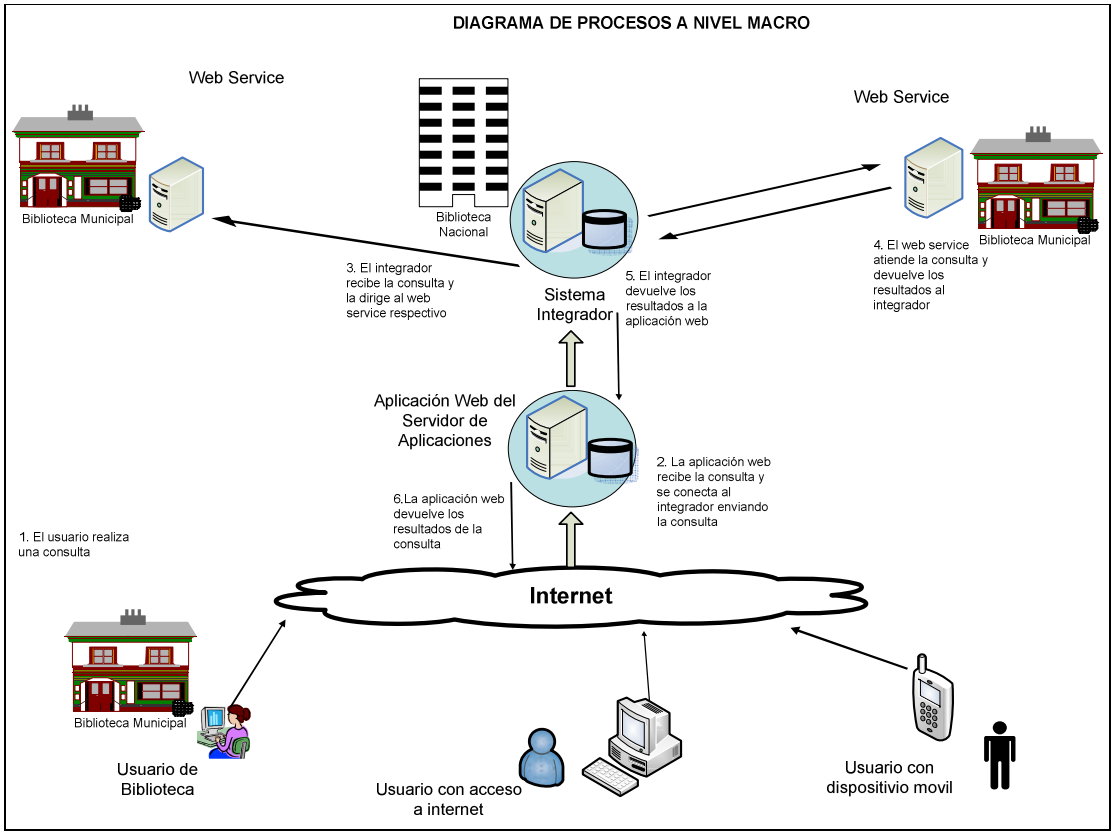


Figura 3.11: Diagrama de Procesos Nivel Macro

DIAGRAMA DE CONTEXTO DE NIVEL 0

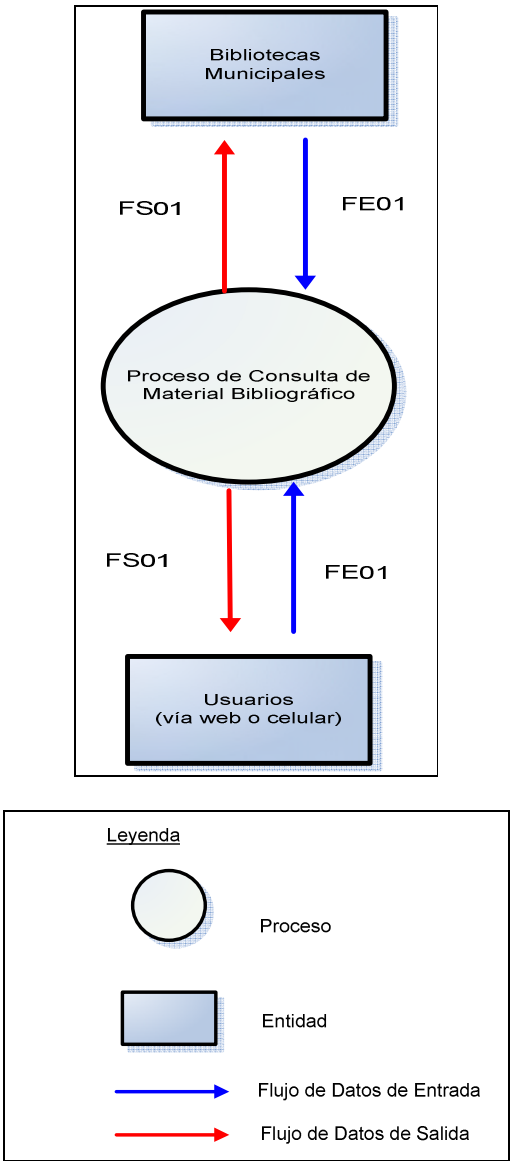


Figura 3.12: Diagrama de Contexto Nivel 0

Del gráfico se tiene:

Entidad	Descripción
Biblioteca Municipal	Encargado de enviar consultas de los usuarios que están dentro del local. En cada biblioteca municipal se tiene un sistema de biblioteca para poder realizar las consultas.
Usuario	Encargado de realizar consultas de material bibliográfico vía web o celular.

Flujo de Entrada	Descripción
FE01	Solicitudes de consulta provenientes de los usuarios y las bibliotecas municipales.

Flujo de Salida	Descripción
FS01	Respuesta a las consultas.

3.4 DESCRIPCIÓN DE FUNCIONES DE SISTEMA

1. Consultar material bibliotecario de una municipalidad seleccionada

El usuario podrá consultar el material bibliográfico (Libros) de una biblioteca seleccionada, esto permitido por el WebServices publicado en la biblioteca y como mediador el ESB (Enterprise Service Bus)

2. Consultar material bibliotecario de todas las municipalidades:

El usuario podrá consultar el material bibliográfico (Libros) de todas las bibliotecas en una sola búsqueda esto permitido por los WebServices publicados en cada biblioteca y como mediador el ESB (Enterprise Service Bus)

3. Consultar material bibliotecario desde un dispositivo móvil.

El usuario podrá consultar el material bibliográfico (Libros) desde un dispositivo móvil haciendo uso de un software J2ME descargado desde un Site e instalado en el dispositivo móvil.

3.5 HERRAMIENTAS A UTILIZAR

Para el desarrollo de la solución se utilizará la plataforma J2EE basado en el lenguaje Java.

Cabe resaltar que las herramientas a utilizar son Open Source, lo que significa ningún costo y con el soporte de la comunidad de usuarios Java y otras tecnologías.

Se utilizaran las siguientes herramientas, clasificadas por los siguientes tipos:

- ✓ Servidor de Aplicaciones
- ✓ Bus de Servicios Empresarial
- ✓ Herramienta de Desarrollo para Java
- ✓ Herramienta de Desarrollo para J2ME
- ✓ Framework de Desarrollo para Web Services
- ✓ Framework para Configuración de Web Services con Mule
- ✓ Framework para desarrollo de Aplicaciones Web

Servidor de Aplicaciones

JBoss : Es un servidor J2EE de código abierto implementado en Java puro, de alto rendimiento. Puede ser implementado en cualquier sistema operativo que soporte Java. La empresa que lo desarrolló fue adquirida por Red Hat en abril del 2006 y está apoyado por una red mundial de colaboradores. Implementa todo el paquete de servicios de la plataforma J2EE. Se utilizará la versión 5.1

Bus de Servicios Empresarial

Mule ESB: Es un framework ligero de mensajería basado en Java que permite rápidamente y fácilmente conectar aplicaciones y habilitarlos para intercambiar datos. Mule utiliza una arquitectura orientada a servicios (SOA) que permite la integración de sistemas existentes. Es altamente escalable e independiente del proveedor. Se utilizará la versión 2.2.1

Herramienta de Desarrollo para Java

Eclipse: Es un entorno de desarrollo integrado de código abierto multiplataforma utilizado para desarrollar aplicaciones java. Eclipse actualmente es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Se utilizará la versión 3.5 o comúnmente llamado Eclipse Galileo.

Herramienta de Desarrollo para J2ME

NetBeans: Es un entorno de desarrollo integrado gratuito de código abierto para desarrollo de software. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento.
Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.
Se utilizará la versión 6.8 que incluye un entorno para J2ME.

Ksoap2: Framework: Librería cliente SOAP WebService para aplicaciones J2ME (CLDC, CDC, MIDP), nos permite comunicarnos con los WebServices. Ksoap maneja perfectamente el manejo de arreglos de objetos, con el fin de recibirlos del webService.
Se utilizará la versión 2.1.2.

Framework de Desarrollo para Web Services

Apache Axis: Es un framework open Source y gratuito para el desarrollo de Web Services basados en XML. Consiste de una implementación en Java y C++ del servidor SOAP, y varias utilidades y Apis para generar y deployar aplicaciones de web services. Se utilizará la versión 1.0

Framework para Configuración de Web Services con Mule

Spring: Es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. El framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. ***(Consultar Referencia Libros [3])***

Framework para desarrollo de Aplicaciones Web

Java Server Faces: Es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en

aplicaciones Java Empresariales. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas. La especificación de JSF fue desarrollada por la Java Community Process. Se utilizará la versión 1.2

Spring: Es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. El framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

CAPÍTULO IV

4. DESARROLLO DEL SISTEMA APLICADO AL CASO DE ESTUDIO

En nuestro caso de estudio comunicaremos consultaremos las bibliotecas de San Isidro y de Miraflores.

4.1 METODOLOGIA APLICADA A LA SOLUCION

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no llevamos una metodología de por medio, lo que obtenemos es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo.

Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí si toma sentido el basarnos en una metodología de desarrollo, y empezamos a buscar cual sería la más apropiada para nuestro caso. Lo cierto es que muchas veces no encontramos la más adecuada y terminamos por hacer o diseñar nuestra propia metodología, algo que por supuesto no está mal, siempre y cuando cumpla con el objetivo.

Muchas veces realizamos el diseño de nuestro software de manera rígida, con los requerimientos que el cliente nos solicitó, de tal manera que cuando el cliente en la etapa final (etapa de prueba), solicita un cambio se nos hace muy difícil realizarlo, pues si lo hacemos, altera muchas cosas que no habíamos previsto, y es justo éste, uno de los factores que ocasiona un atraso en el proyecto y por tanto la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido. Obviamente para evitar estos incidentes debemos haber llegado a un acuerdo formal con el cliente, al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique al desarrollo del mismo.

Muchas veces los usuarios finales, se dan cuenta de las cosas que dejaron de mencionar, recién en la etapa final del proyecto, pese a que se les mostró un prototipo del software en la etapa inicial del proyecto.

Los proyectos en problemas son los que salen del presupuesto, tienen importantes retrasos, o simplemente no cumplen con las expectativas del cliente.
Bajo este contexto hemos seleccionado RUP (Rational Unified Process) como metodología de desarrollo, es una implementación del Desarrollo Espiral. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.
En la siguiente figura se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

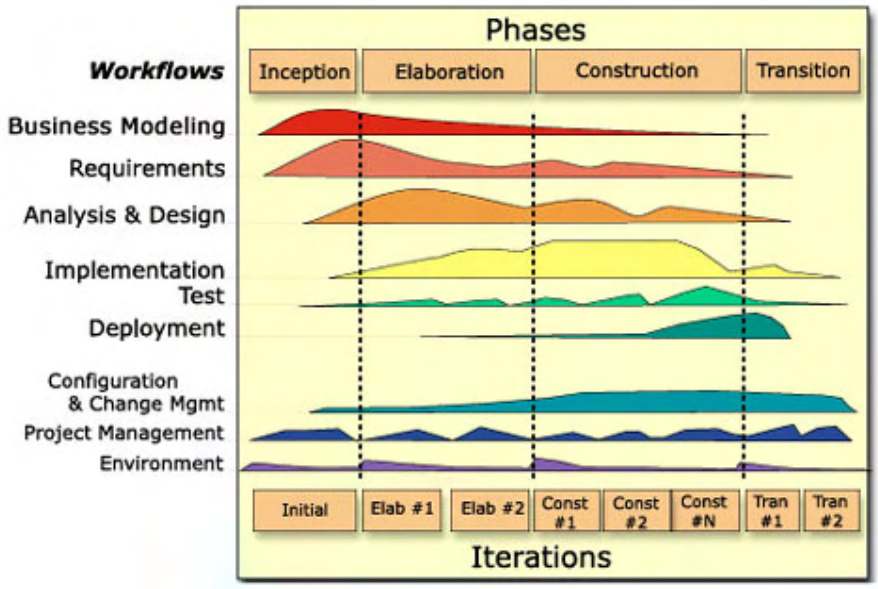


Figura 4.1: Fases del RUP

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una baseline (Línea Base) de la arquitectura.

Durante la fase de ***inicio*** las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos.

En la fase de ***elaboración***, las iteraciones se orientan al desarrollo de la baseline de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la baseline de la arquitectura.

En la fase de ***construcción***, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se selecciona algunos casos de uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.
En la fase de ***transición*** se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

4.2 ANÁLISIS ESTRATÉGICO DEL SISTEMA

4.2.1 ANÁLISIS FODA

Fortalezas

F1 El presupuesto de desarrollo e implantación del sistema no es muy elevado debido a la utilización de software libre.

F2 El sistema brindará un servicio mejorado para las bibliotecas municipales debido a la integración de la información.

F3 El sistema brindará un servicio innovador debido a la posibilidad de acceso mediante un dispositivo móvil.

Debilidades

D1 El sistema solo considera servicios de consulta.

D2 Las fuentes de datos de la información bibliográfica no son óptimas debido a que la mayoría de las bibliotecas municipales cuentan con base de datos a modo de archivos planos.

D3 Impacto en el tiempo de respuesta de las consultas cuando la información bibliográfica aumente, debido a que no se cuentan con las ventajas de una base de datos relacional.

Oportunidades

O1 Beneficios para los usuarios de las bibliotecas municipales.

O2 En el mercado no existen muchas aplicaciones que permiten ser accedidas mediante dispositivos móviles.

O3 La gran cantidad de usuarios portadores de un teléfono móvil y el creciente aumento de adquisición de los mismos.

O4 Se cuentan con las tecnologías necesarias que facilitan el desarrollo del sistema.

O5 Posibilidad de extender la cobertura del sistema a bibliotecas municipales a nivel nacional.

O6 Posibilidad de extender la cobertura del sistema a otros tipos de bibliotecas públicas (universitarias, colegiales, otros).

O7 Brindar un modelo de integración para otras áreas del sector público.

Amenazas

A1 Estimación inexacta de los requerimientos del sistema.

A2 Falta de cooperación por parte de las bibliotecas municipales para facilitar la integración.

A3 Porcentaje de personas con baja cultura tecnológica no pudiendo aprovechar lo que les ofrece.

4.3 REQUERIMIENTOS

4.3.1 RESTRICCIONES Y LÍMITES DEL SISTEMA

Ítem	Descripción de la restricción y el límite	Afecta a…
1	Se utilizarán productos y tecnologías que sean compatibles con la plataforma J2EE.	Todo el sistema
2	Los servicios web serán implementados en el lenguaje Java utilizando el framework Axis.	Todo el sistema
3	El aplicativo web junto con el integrador ESB residirán en un solo host.	Todo el sistema
4	Las bases de datos de información serán en su mayoría del tipo Winisis (archivos planos).	Todo el sistema
5	Se integrarán solamente las bibliotecas municipales de Lima y Callao.	Todo el sistema

4.3.2 REQUERIMIENTOS FUNCIONALES

Código	Descripción	Tipo	Prioridad	Criticidad
RQIBM001	Permitir la consulta de material bibliográfico en una biblioteca municipal.	AUC	C	Alta
RQIBM002	Permitir la consulta de material bibliográfico en todas las bibliotecas municipales.	AUC	C	Alta
RQIBM003	Permitir la consulta de material bibliográfico vía dispositivo móvil.	AUC	N	Media

Donde:

Código: RQXXXNNN
XXX: Es el código del Área Usuaria
NNN: Es un secuencial
Prioridad: Crítico (C),
Necesario (N),
Deseable (D)
TIPO:
AUC: Asociados a los casos de uso.
AAG: Asociados a aspectos generales

4.3.3 REQUERIMIENTOS NO FUNCIONALES

Código	Descripción	Tipo	Prioridad	Criticidad
RQIBM001	El software a desarrollar debe tener la capacidad de ser comprendido, aprendido, usado y atractivo para el usuario.	USA	D	Media
RQIBM002	Mejorar la imagen y prestigio, buscando optimizar los costos de diseño, rediseño y mantenimiento.	USA	D	Media
RQIBM003	Se buscará que el sistema tenga la fiabilidad necesaria que garantice el buen funcionamiento del sistema.	CON	C	Alta
RQIBM004	El sistema estará disponible como mínimo del 90% entre las 8:00 a.m. y las 08:00 p.m.	CON	C	Alta
RQIBM005	El sistema estará disponible 24x365 días al año.	CON	C	Alta
RQIBM006	Se garantizará que la información emitida será el fiel reflejo del contenido de la base de datos.	CON	C	Alta
RQIBM006	Se buscará que el sistema tenga los niveles mínimos de rendimiento al recuperar y guardar la información. Pero la mayor parte del rendimiento será cubierta por el hardware, software y acceso de comunicación del servidor en donde resida el sistema.	REN	C	Alta

Donde:

Código: RQXXXXNNN
XXXX: Es el código del Área Usuaria
NNN: Es un secuencial

Prioridad: Crítico (C),
Necesario (N),
Deseable (D)

TIPO:
USA: Usabilidad.
CON: Confiabilidad
REN: Rendimiento
SOP: Soporte.

RED: Restricciones de Diseño
DUA: Documentación de Usuario y Sistema de Ayuda
COA: Componentes Adquiridos
INU: Interfases de Usuarios.
INH: Interfases de Hardware
INS: Interfases de Software
INC: Interfases de Comunicaciones
LIC: Licenciamiento
RLE: Requerimientos Legales y de Derecho de Autor
RUP: Estándares Aplicables.]

4.4 ANÁLISIS Y DISEÑO

4.4.1 ESPECIFICACIONES DEL CASO DE USO

Referencia Vista de Caso de Uso del Modelo de Arquitectura del Software 4+1 Vistas

4.4.2 DIAGRAMA DE CLASES

Referencia Vista Lógica del Modelo de Arquitectura del Software 4+1 Vistas

4.4.3 DIAGRAMA DE SECUENCIA

Diagrama de Secuencia de consulta a la Biblioteca de San Isidro

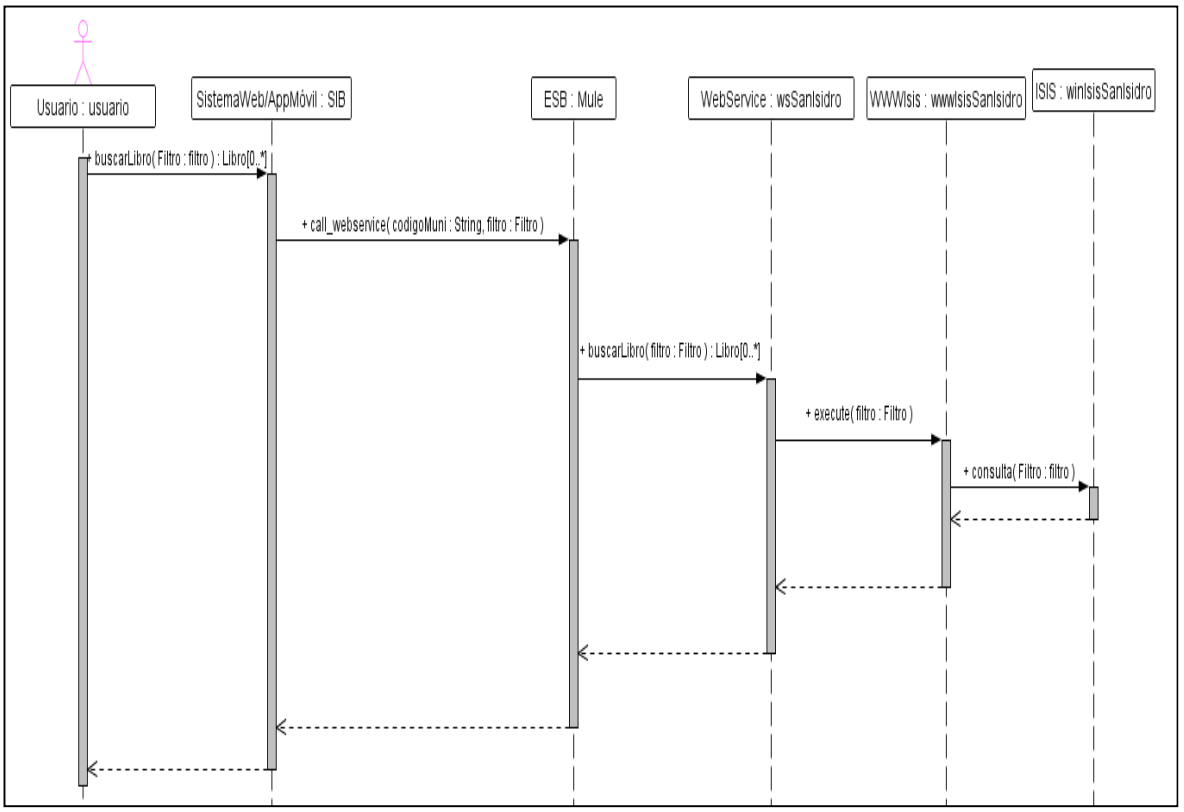


Figura 4.2: Diagrama de Secuencia Biblioteca San Isidro

Diagrama de Secuencia de consulta a la Biblioteca de Miraflores

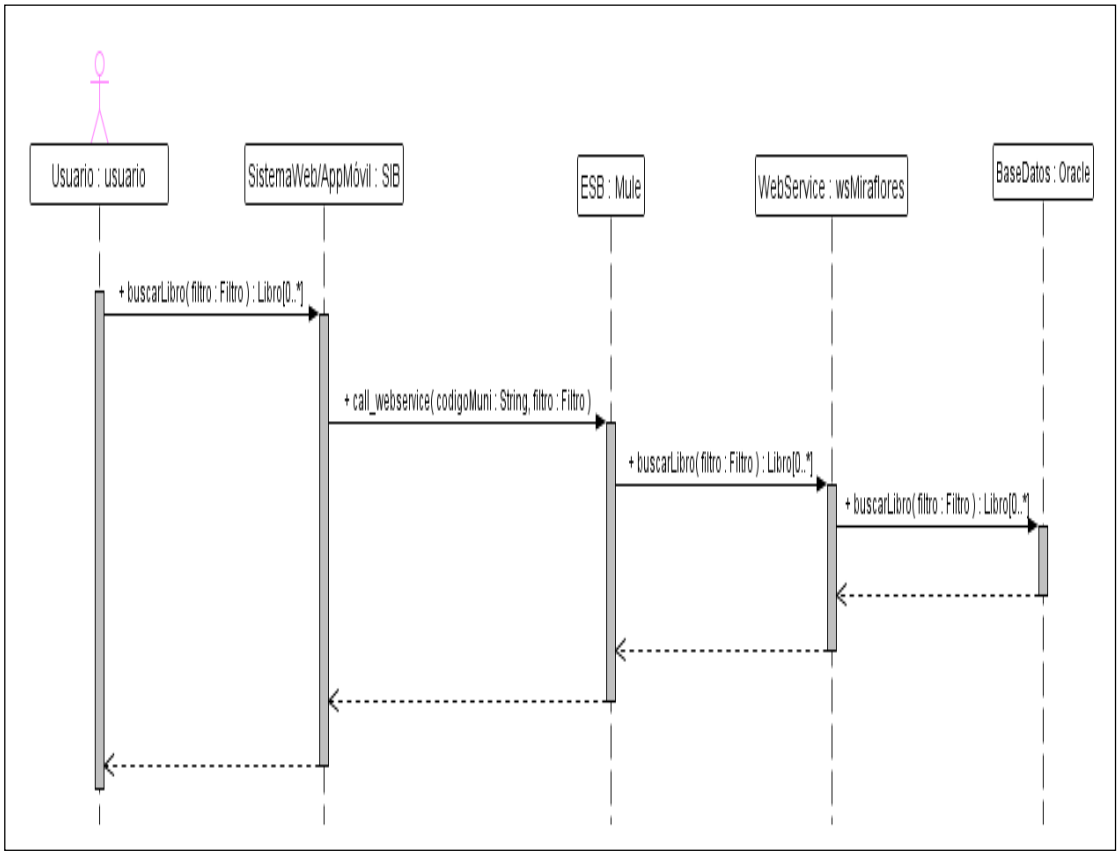


Figura 4.3: Diagrama de Secuencia Biblioteca de Miraflores

4.5 CONFIGURACIÓN Y PRINCIPALES CONEXIONES

En esta parte vamos a configurar y realizar las principales conexiones de la aplicación, del ESB (Mule) de la aplicación Web y de la aplicación Móvil

4.5.1 CONFIGURACIÓN DEL ESB USANDO MULE

MUe encargado de rutear las peticiones hacia el Web Service correspondiente.

1. Descargarmos el Mule de la página http://www.mulesof.org la versión 2.2.1
2. Lo descomprimimos en una carpeta local
3. Verificamos el JAVAHOME y agregamos la variable de entorno MULE_HOME, que debe apuntar el directorio donde se encuentra el Mule
4. Añador $MULE_HOME/bin al PATH

Ahora crearemos el archivo configbib.xml de configuración para el MULE

```
<spring:bean name="WSProxyService" class="org.mule.transport.soap.WSProxyService">
    <spring:property name="wsdlFile" value="c:/xml/Biblioteca.wsdl"/>
</spring:bean>

<model name="webServiceProxy">
   <service name="HttpWebServiceBridge_Miraflores">
       <inbound>
            <inbound-endpoint address="http://pcMule:8080/12" synchronous="true"/>
       </inbound>
       <component>
            <spring-object bean="WSProxyService" />
       </component>
       <outbound>
           <chaining-router>
               <outbound-endpoint address="http://serverMiraflores:8080/BibliotecaMiraflores/services/Biblioteca" synchronous="true"/>
           </chaining-router>
       </outbound>
    </service>

    <service name="HttpWebServiceBridge_SanIsidro">
       <inbound>
            <inbound-endpoint address="http://pcMule:8080/15 " synchronous="true"/>
       </inbound>
       <component>
            <spring-object bean="WSProxyService" />
       </component>
       <outbound>
           <chaining-router>
               <outbound-endpoint address="http://serverSanIsidro:8080/BibliotecaSanIsidro/services/Biblioteca" synchronous="true"/>
           </chaining-router>
       </outbound>
    </service>
   </model>
</mule>
```

Las peticiones de las aplicaciones Web o J2ME ingresan en este "End Point" apuntando al servidor Mule e identificando a la biblioteca municipal mediante una etiqueta, en este caso la etiqueta "12" que es el código de la municipalidad.

La petición es enviada por este "End Point" hacia el WebService correpondiente

Este segundo bloque es para la biblioteca de San Isidro con código "15", y así sucesivamente con las demás bibliotecas

Figura 4.4: Archivo de configuración del MULE

Notar que el Mule lee este archivo de congiruación , y es a través de él que el Mule conoce donde se encuentra las direcciones de los WebServices de acuerdo al código de municipalidad.

4.5.2 LLAMADA AL WEBSERVICE DESDE EL SISTEMA INTEGRADOR DE BIBLIOTECAS

Invocamos al WebService del Sistema de Biblioteca de la Municipalidad de Miraflores mediante el cliente generado.

Notar que la aplicación se conecta al ESB mediante la IP del ESB, en este caso la IP del servidor donde está corriendo el Mule, que es el encargado de conectar con el WebService correspondiente según el código de municipalidad

En este método enviamos una petición al MULE que deseamos conectarnos con el WebService de la municipalidad de Miraflores, y llamar al método “buscarLibro”.

```
public  Libro[] buscarLibro(String codigoMuni, String opcion, String
criterio)throws Exception{

BibliotecaServiceLocator locator= new
BibliotecaServiceLocator();
//http://serverMule:8080/12
locator.setBibliotecaEndpointAddress(IP_MULE+codigoMuni)
Biblioteca biblioteca=locator.getBiblioteca();
Libro lib[]=biblioteca.buscarLibros(opcion, criterio);//Método
a //llamar

return lib;


}
```

4.5.3 LLAMADA AL WEBSERVICE DESDE LA APLICACIÓN J2MEE

Notar que la aplicación se conecta al ESB mediante la IP del ESB, en este caso la IP del servidor donde está corriendo el Mule, que es el encargado de conectar con el WebService correspondiente según el código de municipalidad.

En este método enviamos una petición al MULE que deseamos conectarnos con el WebService de la municipalidad de Miraflores, y llamar al método “buscarLibro”.

```
public  Libro[] buscarLibro(String codigoMuni, String opcion, String
criterio)throws Exception{

String url=BibliotecaSingleton.getInstance().getIpEsb()+codigoMuni;

//url="http://serverMule:8080/12";

SoapObject client=new SoapObject(url,"buscarLibros");//Método a
llamar
client.addProperty("opcion",opcion);//variable enviada
client.addProperty("criterio",criterio);//variable enviada

SoapSerializationEnvelope envelope=new
SoapSerializationEnvelope(SoapEnvelope.VER11);

envelope.setOutputSoapObject(client);
HttpTransport ht=new HttpTransport(url);
ht.call("buscarLibros", envelope);//Enviamos el mensaje

SoapObject result = (SoapObject)envelope.bodyIn;//Resultado

//Pasamos la respuesta del mensaje a un arreglo
Libro lib[]=null;
if(result.getPropertyCount()>0){
       lib=new Libro[result.getPropertyCount()];
       SoapObject obj;
       for(int i=0;i<lib.length;i++){
           obj=(SoapObject)result.getProperty(i);
           lib[i]=new Libro();
           lib[i].setCodigo(obj.getProperty("codigo").toString());
           lib[i].setTitulo(obj.getProperty("titulo").toString());
           lib[i].setAutor(obj.getProperty("autor").toString());
           lib[i].setTema(obj.getProperty("tema").toString());

lib[i].setBiblioteca(obj.getProperty("biblioteca").toString());
       }
   }
return lib;//Devolvemos el Arreglo
  }
```

4.5.4 EJECUCIÓN DEL MULE

Para ejecutar el Mule ingresamos en una ventana de comandos ingresamos lo siguiente:

```
C:\mule -config c:\xml\configbib.xml
```

Donde configbib.xml es el archivo de configuración del Mule.

Una vez levantado el servicio obtendremos la siguiente pantalla, listo para esperar las peticiones que lleguen,

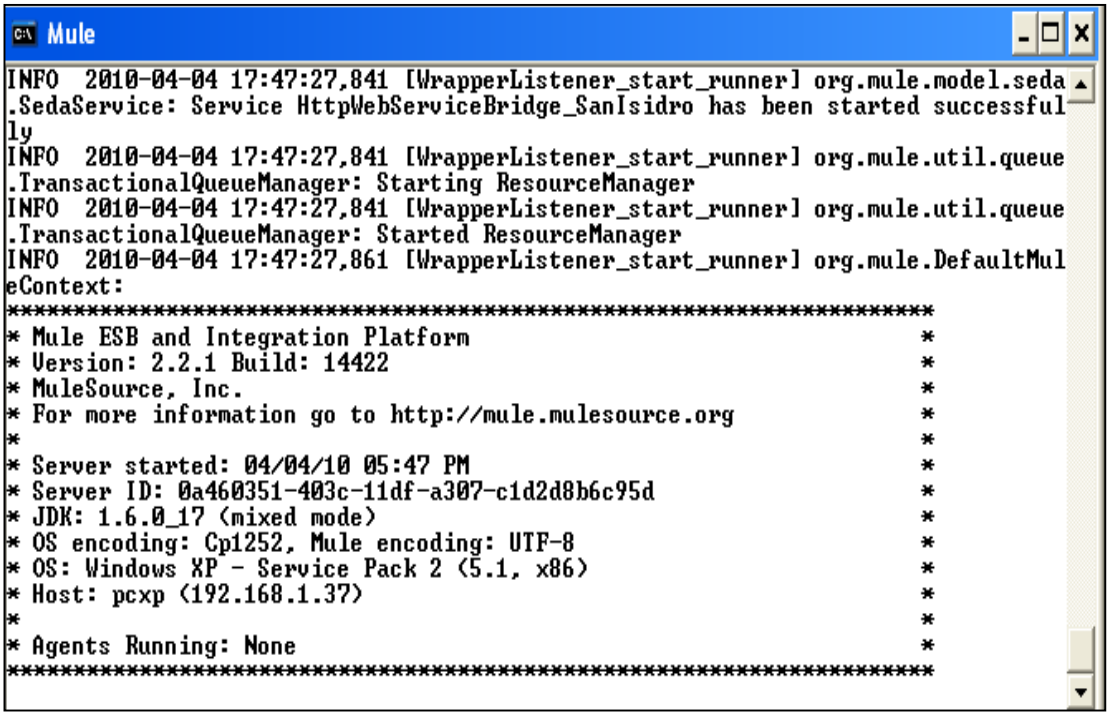


Figura 4.5: Consola del Mule

4.6 ESPECIFICACIONES TECNICAS

4.6.1 REQUISITOS DE HARDWARE

El hardware necesario para la implementación de la solución constará de lo siguiente:

- Para el servidor donde se alojara la aplicación web y el integrador ESB se debe cumplir los siguientes requisitos:

Componente	Requisitos Mínimos	Recomendación
Procesador	Pentium Core 2 Duo 32 bits 3,00 Ghz	Pentium Core 2 Duo 64 bits 3,33 Ghz
Memoria RAM	4Gb	8Gb
Disco Duro	250 Gb	250Gb
Espacio Libre en el Disco Duro	10Gb	10Gb

- Para los usuarios que cuenten con dispositivos móviles:

Componente	Requisitos Mínimos	Recomendación
Dispositivo Móvil	Capacidad de ejecutar aplicaciones Java	Capacidad de ejecutar aplicaciones Java

4.6.2 REQUISITOS DE SOFTWARE

El software que se utilizará para alojar la implementación de solución en el servidor es el siguiente:

- Sistema Operativo Windows NT, Windows 2000, Windows 2003 o Unix.
- El kit de desarrollo de software de java (JDK) versión 1.5 o superior.
- Servidor de Aplicaciones JBoss versión 5.1.
- El bus de servicios Mule ESB.

El software que se utilizará en las bibliotecas municipales para la publicación de sus web services es el siguiente:

- Servidor de Aplicaciones propia de la biblioteca.
- Base de datos WinIsis u otra de la biblioteca.
- Aplicativo WWWIsis o Web de la biblioteca.

4.6.3 REQUISITOS DE PROCESO

Para que el sistema y los usuarios puedan interactuar exitosamente se deben tener en cuenta los siguientes requisitos:

- El servidor debe estar encendido en todo instante.
- Los servicios web de cada municipalidad deben estar funcionando en todo momento.
- Las bases de datos deben estar activas en todo instante.
- El servidor de aplicaciones debe estar ejecutándose en todo momento.

4.7 PROTOTIPOS

4.7.1 APLICACIÓN WEB (SISTEMA INTEGRADOR DE BIBLIOTECAS)

Pantalla Inicial

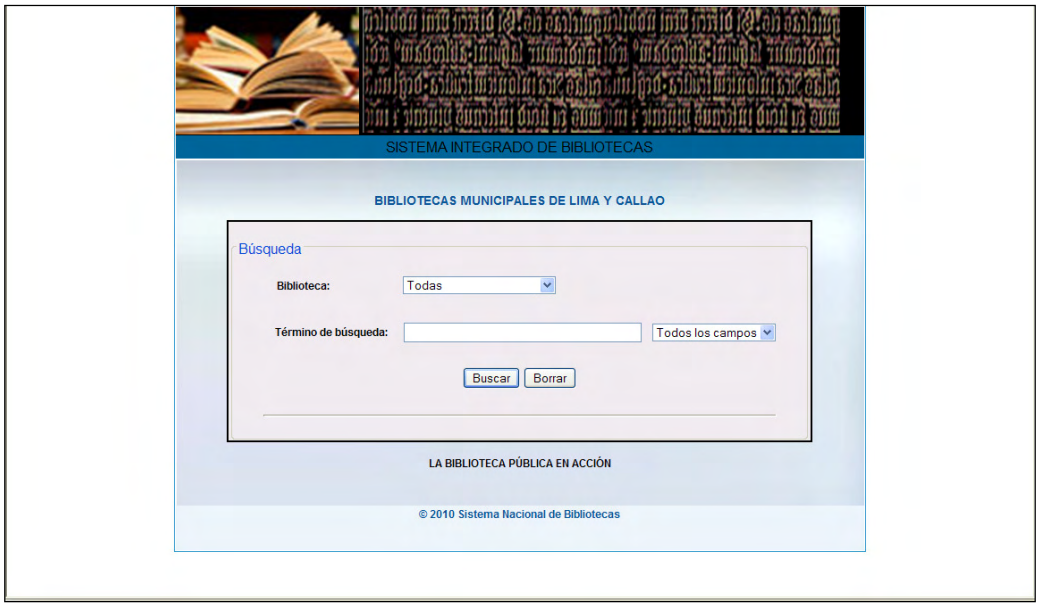


Figura 4.6: Pantalla Inicial

Búsqueda Por Biblioteca Municipal

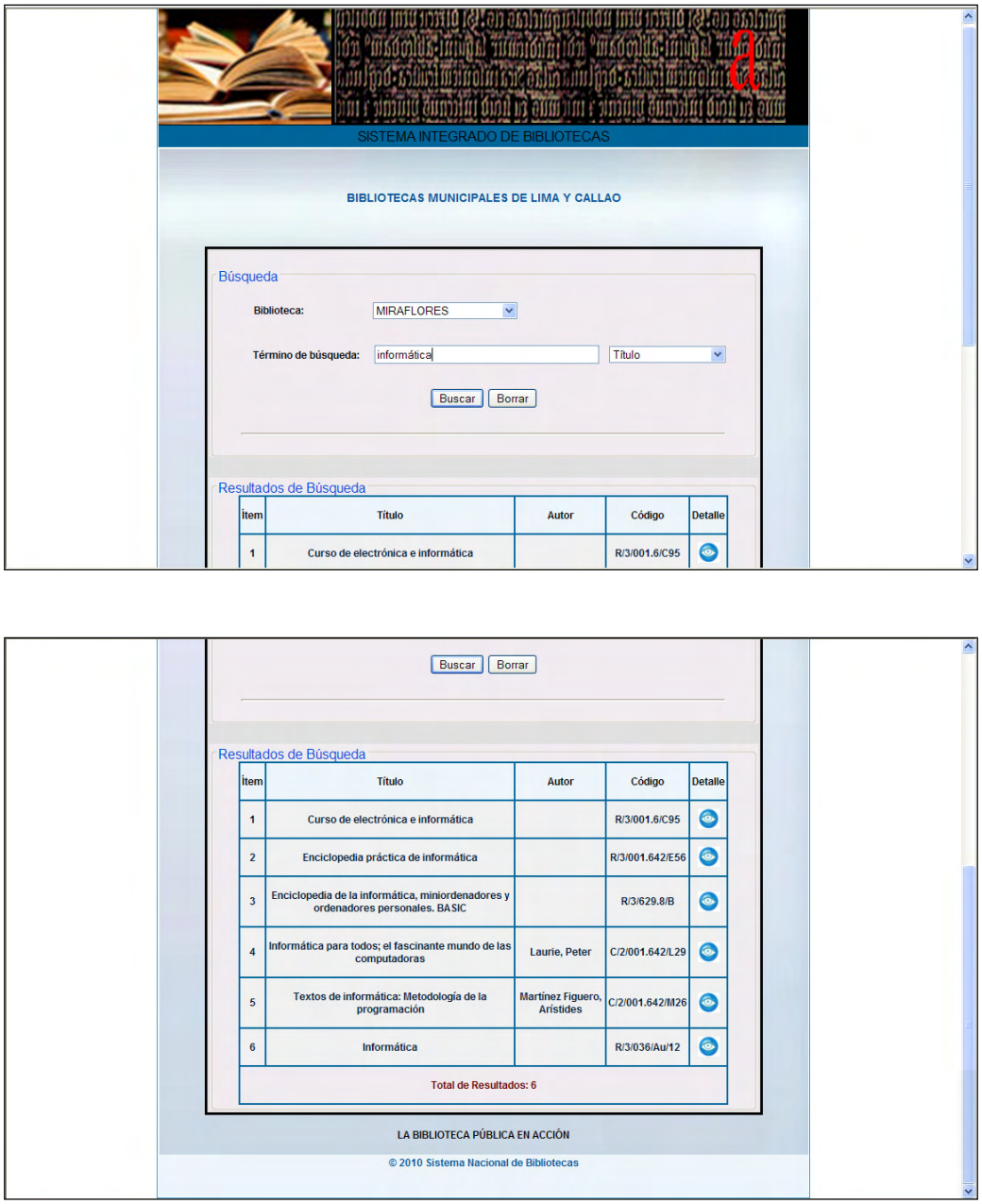


Figura 4.7: Búsqueda por Biblioteca

Búsqueda Por todas las Bibliotecas Municipales

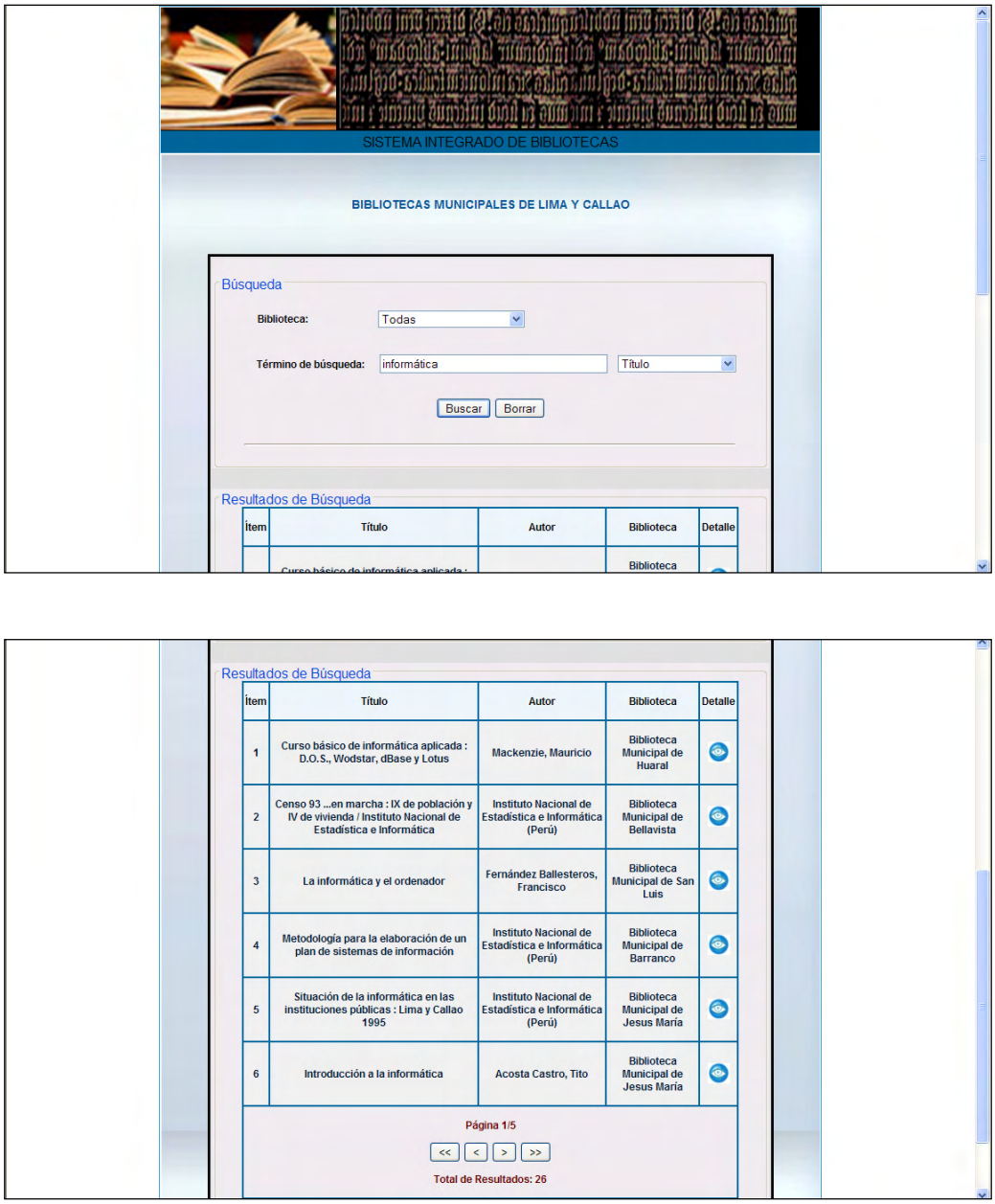


Figura 4.8: Búsqueda por todas las Bibliotecas

Detalle

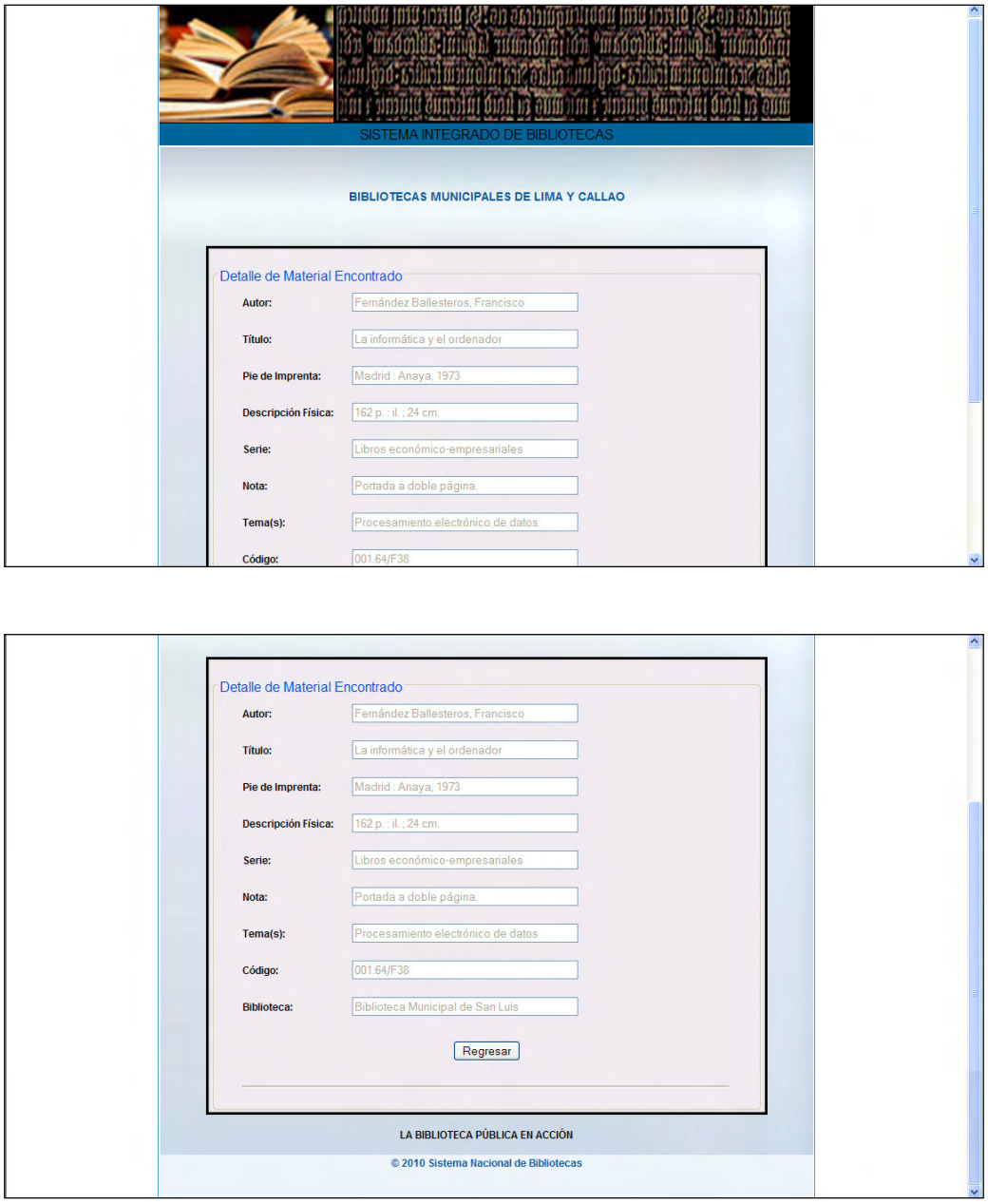


Figura 4.9: Detalle de una Biblioteca

4.7.2 APLICACIÓN MÓVIL

1. Pantalla de Bienvenida

Figura 4.10: Pantalla de Bienvenida

2. Elegimos una Biblioteca Municipal, o todas las bibliotecas municipales y presionamos el botón Aceptar

Figura 4.11: Selección de Biblitoeca

3. Nos presenta 3 opciones de búsqueda, elegimos por Autor y escribimos en el camp criterio “Mario Vargas Llosa”, seguidamente escogemos Menú – Buscar.

Figura 4.12: Selección de opciones de Búsqueda

4. El Sistema procesa la petición y devuelve los resultados:

Figura 4.13: Proceso y Resultado de búsqueda

5. Usando las flechas direccionales del dispositivo móvl nos podemos mover por los demás campos de la tabla.

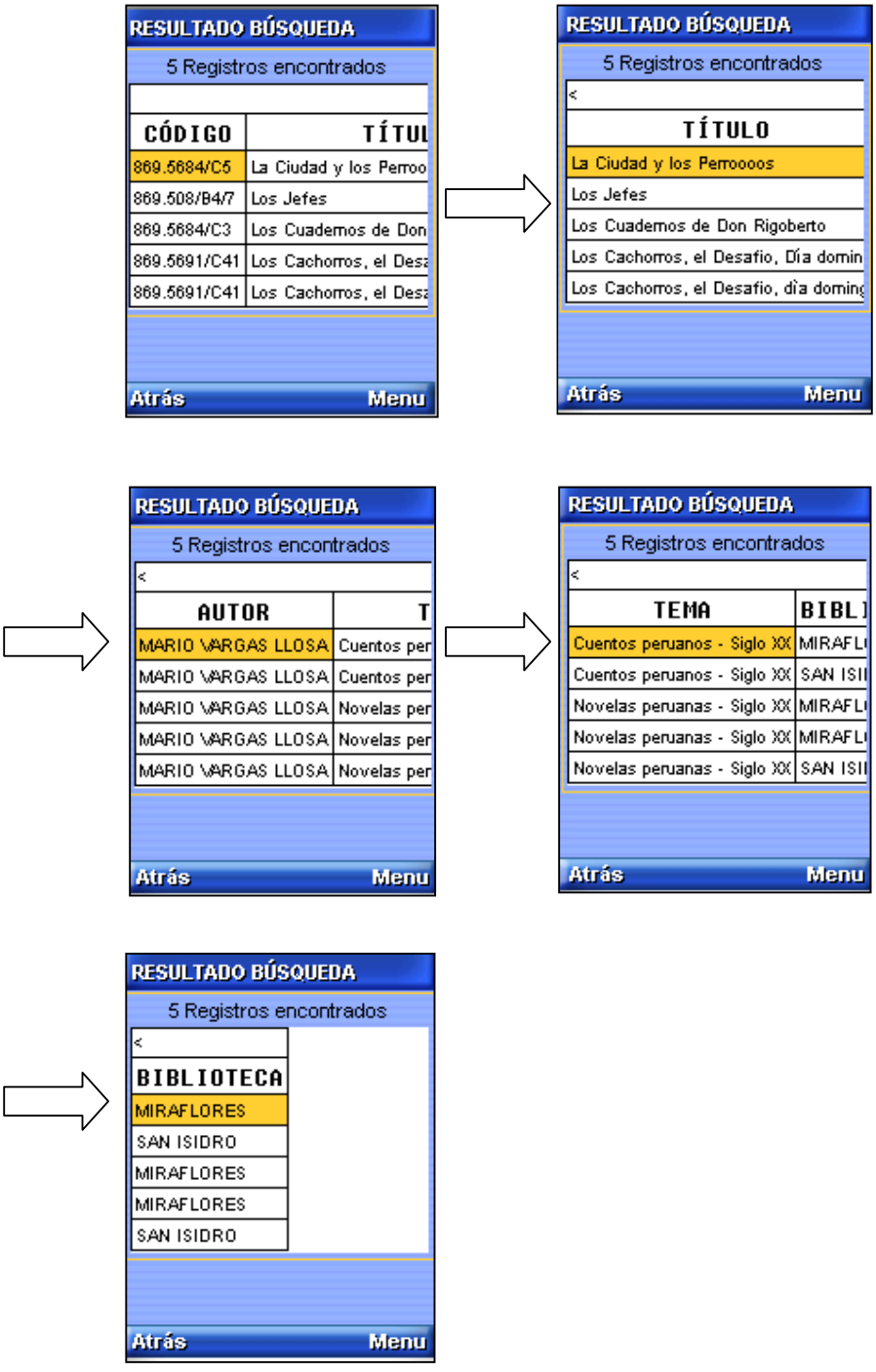


Figura 4.13: Visualizar pantalla de Resultados

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

1. En el presente trabajo hemos analizado una posible solución de la problemática actual en las Bibliotecas Municipales referente a la integración, mediante la elaboración de un planteamiento que propone una Arquitectura Orientada a Servicios, la utilización de Web Services para definir cada servicio, la implantación de un Bus de Servicios Empresarial para llevar a cabo la integración y la posibilidad de acceder a esta solución mediante un dispositivo móvil.

2. Se concluye que la implementación de esta solución permitirá definitivamente la integración de información de las bibliotecas municipales de Lima y Callao.

3. A través de la implementación de esta solución se brindará un mejor servicio a los usuarios de las bibliotecas municipales facilitándoles el acceso a la información del material bibliográfico, contribuyendo en algo a la investigación y educación de la población.

5.2 RECOMENDACIONES

1. Como recomendación se puede tomar este modelo de integración, para las bibliotecas universitarias o cualquier tipo de biblioteca, pudiendo abarcar un ámbito nacional.

2. Para las Bibliotecas municipales que no cuentan aún con un sistema de Bibliotecas, se recomienda implantar una base de datos WinISIS que es gratuito y después integrarlo al WWWISIS para que puedan hacerse búsquedas desde la WEB. De esta manera se estaría dando un previo paso para su publicación de sus servicios (WebServices), preparándose para la integración con el Sistema Integrador de Bibliotecas Municipales.

CAPÍTULO VI

6. GLOSARIO DE TÉRMINOS Y ACRÓNIMOS

1 **SOA:** *Service Oriented Architecture*, es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio. Proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

2 **ESB:** *Enterprise Service Bus*. Consiste en un combinado de arquitectura de software que proporciona servicios fundamentales para arquitecturas complejas a través de un sistema de mensajes (el bus) basado en las normas y que responde a eventos. Los desarrolladores normalmente implementan un BSE utilizando tecnologías de productos de infraestructura de middleware que se basan en normas reconocidas.

3 **J2ME:** *Java 2 Micro Edition.* Es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrollo de software para dispositivos con recursos restringidos. Está orientado a productos de consumo como PDAs, teléfonos móviles o electrodomésticos.

4 **OASIS:** *Organization for the Advancement of Structured Information Standards*. Es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares de comercio electrónico y servicios web.

5 **W3C:** *El World Wide Web Consortium*, abreviado W3C, es un consorcio internacional que produce recomendaciones para la World Wide Web.

6 **WS-I:** Dentro de la arquitectura SOA para la implementación de Servicios Web, la interoperabilidad es tal vez el principio más importante. Como método de implementación de SOA, Web Services debe ofrecer importantes beneficios de interoperabilidad, y permitir la ejecución de servicios Web distribuidos en múltiples plataformas de software y arquitecturas de hardware.

7 **XML:** Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

8 **SOAP:** *Simple Object Access Protoco* **.**Protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

9 **XML-RPC:** Protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

Es un protocolo muy simple ya que solo define unos cuantos tipos de datos y comandos útiles, además de una descripción completa de corta extensión. La simplicidad del XML-RPC está en contraste con la mayoría de protocolos RPC que tiene una documentación extensa y requiere considerable soporte de software para su uso.

10 **HTTP:** *Hypertext Transfer Protocol.* Protocolo de transferencia de hipertexto. Es el protocolo usado en cada transacción de la World Wide Web. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web o un spider) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

11 **FTP:** *File Transfer Protocol* - Protocolo de Transferencia de Archivos. En informática, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado

en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

12 **SMTP:** *Simple Mail Transfer Protocol.* Protocolo Simple de Transferencia de Correo, es un protocolo de la capa de aplicación. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.). Es un estándar oficial de Internet

13 **WSDL:** Son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

14 **UDDI:** Son las siglas del catálogo de servicios web de Internet denominado *Universal Description, Discovery and Integration*. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. El registro de un negocio en UDDI tiene tres partes:

Páginas blancas - dirección, contacto y otros identificadores conocidos.
Páginas amarillas - categorización industrial basada en taxonomías.
Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

15 **FIREWALL:** Un muro de fuego (***firewall*** en inglés) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar, descifrar, el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios.

Los cortafuegos pueden ser implementados en hardware o software, o una combinación de ambos. Los cortafuegos se utilizan con frecuencia para evitar que los usuarios de Internet no autorizados tengan acceso a redes privadas conectadas a Internet, especialmente intranets. Todos los mensajes que entren o salgan de la intranet pasan a través del cortafuegos, que examina cada mensaje y bloquea aquellos que no cumplen los criterios de seguridad especificados. Un cortafuegos correctamente configurado añade una protección necesaria a la red, pero que en ningún caso debe considerarse suficiente.

16 **CORBA:** *Common Object Request Broker Architecture* — Arquitectura común de intermediarios en peticiones a objetos, es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CORBA fue definido y está controlado por el *Object Management Group* (OMG) que define las APIs, el protocolo de comunicaciones y los mecanismos necesarios para permitir la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida.

En un sentido general, CORBA "envuelve" el código escrito en otro lenguaje, en un paquete que contiene información adicional sobre las capacidades del código que contiene y sobre cómo llamar a sus métodos. Los objetos que resultan, pueden entonces ser invocados desde otro programa (u objeto CORBA) desde la red. En este sentido CORBA se puede considerar como un

formato de documentación legible por la máquina, similar a un archivo de cabeceras, pero con más información.

CORBA utiliza un lenguaje de definición de interfaces (IDL) para especificar las interfaces con los servicios que los objetos ofrecerán. CORBA puede especificar a partir de este IDL, la interfaz a un lenguaje determinado, describiendo cómo los tipos de dato CORBA deben ser utilizados en las implementaciones del cliente y del servidor. Implementaciones estándar existen para Ada, C, C++, Smalltalk, Java, Python, Perl y Tcl.

17 **RMI** *Java Remote Method Invocation*. Mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y provee de un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI.

18 **DCOM** *Distributed Component Object Model,* Modelo de Objetos de Componentes Distribuidos, es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí. Extiende el modelo COM de Microsoft y proporciona el sustrato de comunicación entre la infraestructura del servidor de aplicaciones COM+ de Microsoft. Ha sido abandonada en favor del framework .NET0

19 **EDI** *Electronic Data Interchange. Intercambio Electrónico de datos* o *EDI*, es un software middleware que permite la conexión a distintos sistemas empresariales como ERP o CRM. El intercambio electrónico de datos puede realizarse en distintos formatos: EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport), XML, ANSI ASC X12, TXT, entre otros.

20 **RPC** *Remote Procedure Call*, Llamada a Procedimiento Remoto Protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el

programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

21 **J2EE** *Java Platform, Enterprise Edition* o **Java EE** (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una *especificación*. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son *conformes a Java EE*; estandarizado por The Java Community Process / JCP.

22 **URL** *Uniform Resource Locator* Un localizador uniforme de recursos, más comúnmente denominado URL es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos, presentaciones digitales, etcétera.

23 **UBR** *Unespecified Bit Rate:* Tasa de velocidad no especificada. Utilizado en el ancho de banda restante. El tráfico que utiliza este servicio es el susceptible de ser eliminado en caso de congestión en los conmutadores. Lo utilizan aplicaciones tolerantes a pérdidas de paquetes, como conexiones TCP.

24 **TCP** *Transmission Control Protocol,* Protocolo de Control de Transmisión, es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 y 1974 por Vint Cerf y Robert Kahn.

Muchos programas dentro de una red de datos compuesta por computadoras pueden usar TCP para crear *conexiones* entre ellos a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que

se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

25 IETF *Internet Engineering Task Force*, Grupo de Trabajo en Ingeniería de Internet. Es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Fue creada en EE.UU. en 1986. La IETF es mundialmente conocida por ser la entidad que regula las propuestas y los estándares de Internet, conocidos como RFC.

Es una institución sin fines de lucro y abierta a la participación de cualquier persona cuyo objetivo es velar porque la arquitectura de Internet y los protocolos que la conforman funcionen correctamente. Se la considera como la organización con más autoridad para establecer modificaciones de los parámetros técnicos bajo los que funciona la red. La IETF se compone de técnicos y profesionales en el área de redes, tales como investigadores, diseñadores de red, administradores, vendedores, entre otros.

26 JMS *Java Message Service* . S*ervicio de mensajes Java*, Es la solución creada por Sun Microsystems para el uso de colas de mensajes. Este es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes. También hace posible la comunicación confiable de manera síncrona y asíncrona.

El servicio de mensajería instantánea también es conocido como *Middleware Orientado a Mensajes* (MOM por sus siglas en inglés) y es una herramienta universalmente reconocida para la construcción de aplicaciones empresariales.

27 REST *Representational State Transfer.* Transferencia de Estado Representacional. Es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término *REST* se refería originalmente a un conjunto de principios de arquitectura —descritos más abajo—, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST de Fielding y también es posible diseñar interfaces XMLHTTP de acuerdo con el estilo de llamada a procedimiento remoto pero sin usar SOAP.

28 **WEB SERVICE** Conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

29 **HTML** *HyperText Markup Language. Lenguaje de Marcado de Hipertexto*, es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

30 **MIME** *Multipurpose Internet Mail Extensions*. Extensiones Multipropósito de Correo de Internet. Consiste en una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario. Una parte importante del MIME está dedicada a mejorar las posibilidades de transferencia de texto en distintos idiomas y alfabetos.

31 WAP *Wireless Application Protocol* . Protocolo de aplicaciones inalámbricas. Es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

Se trata de la especificación de un entorno de aplicación y de un conjunto de protocolos de comunicaciones para normalizar el modo en que los dispositivos inalámbricos, se pueden utilizar para acceder a correo electrónico, grupo de noticias y otros.

32 WML *Wireless Markup Language.* Es un lenguaje cuyo origen es el XML (eXtensible Markup Language). Este lenguaje se utiliza para construir las páginas que aparecen en las pantallas de los teléfonos móviles y los asistentes personales digitales (PDA) dotados de tecnología WAP. Es una versión reducida del lenguaje HTML que facilita la conexión a Internet de dichos dispositivos y que además permite la visualización de páginas web en dispositivos inalám bricos que incluyan la tecnología WAP.

CAPÍTULO VII

7. REFERENCIAS BIBLIOGRÁFICAS

7.1 LIBROS

[1] David J. Coloma Santibáñez (2004). “La Biblioteca Pública en el Perú“. Lima.

[2] Agustín Froute Quintas, Patricia Jorge Cárdenas (2004). “J2ME Java 2 Micro Edition Manual de Usuario y tutorial”. España. Editorial RAM-A.

[3] Craig Walls, Ryan Breidenbach (2005). “Spring in Action“. Estados Unidos. Manning Publication.

7.2 URL

[1] Manifiesto de la UNESCO sobre la Biblioteca Pública 1949
http://www.bnp.gob.pe/bib_publicas/pdf/Manifiesto_Unesco_1949.pdf

[2] Manifiesto de la UNESCO sobre la Biblioteca Pública 1972
http://www.bnp.gob.pe/bib_publicas/pdf/Manifiesto_Unesco_1972.pdf

[3] Manifiesto de la UNESCO sobre la Biblioteca Pública 1994
http://www.bnp.gob.pe/bib_publicas/pdf/Manifiesto_Unesco_BP_1994.pdf

[4] Mule 2 Getting Started Guide
http://www.mulesoft.org/display/MULE2INTRO/Home

[5] Architecture blueprints—the “4+1” view model of software architecture (Philippe Kruchten, 1995)
http://portal.acm.org/citation.cfm?id=216591.216611

[6] Arquitectura orientada a servicios
http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios (2010, abril, 7)

[7] Blog de Arquitectura Orientada a Servicios (SOA)
http://arquitecturaorientadaaservicios.blogspot.com/2006/06/principios-de-la-orientacin-servicios.html

[8] Service Oriented Architecture (SOA)
http://www-01.ibm.com/software/solutions/soa/

[9] Arquitectura orientada a servicios (SOA)
http://cl.sun.com/practice/software/soa/

[10] La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real
http://download.microsoft.com/download/c/2/c/c2ce8a3a-b4df-4a12-ba18-7e050aef3364/070717-Real_World_SOA.pdf

[11] Service Oriented Architecture
http://en.wikipedia.org/wiki/Service-oriented_architecture

[12] Enterprise application integration
http://es.wikipedia.org/wiki/Enterprise_application_integration

[13] Enterprise Integration -- EAI vs. SOA vs. ESB
http://www.ebizq.net/white_papers/6912.html

[14] Service-Oriented Architecture Implementation Framework
http://www.soablueprint.com/whitepapers/SOAPGPart2.pdf

[15] What Is SOA? An Introduction to Service-Oriented Computing
http://www.whatissoa.com/

[16] Principios de la Orientación a Servicios
http://tecnoblog.entel.es/?p=23

[17] IBM – Web Services
http://www.ibm.com/developerworks/webservices/newto/

[18] World Wide Web Consortium (W3C)
http://www.w3.org/

[19] Web Services Interoperability Organization (WS-I)
http://www.ws-i.org/

[20] Simple Object Access Protocol (SOAP)
http://www.w3.org/2000/xp/Group/

[21] Web Services Description Language (WSDL)
http://www.w3.org/2002/ws/desc/

[22] Microsoft Web Services
http://www.desarrollaconmsdn.com/msdn/ServiciosWeb/default.aspx

[23] SOAP and performance
www-106.ibm.com/developerworks/library/ws-quality.html

[24] ¿Qué son los Servicios Web XML?
http://www.netscum.dk/latam/net/basics/xmlservices.asp

[25] Wolter, Roger
2003 Fundamento de los Servicios Web XML.
http://www.microsoft.com/spanish/msdn/articulos/archivo/151102/voices/fundamentos_xml.asp

[26] 2006 Integración y Servicios Web.
http://www.kalysis.com/content/modules.php?op=modload&name=EasyContent&file=index&menu=1500&page_id=12

[27] W3C: Web Services Activity, W3C Draft,
http://www.w3.org/2002/ws/

[28] UDDI, Introduction to UDDI, OASIS. Octubre, 2004.
http://uddi.org/pubs/uddi-tech-wp.pdf (2006, Mayo, 21)

[29] Representational State Transfer
http://en.wikipedia.org/wiki/Representational_State_Transfer

[30] RESTful Web Services
http://java.sun.com/developer/technicalArticles/WebServices/restful/

[31] RESTful Web services: The basics
https://www.ibm.com/developerworks/webservices/library/ws-restful/

[32] Web Services REST
http://sunopencommunitiesforum.es/pdfs_agenda/track_1_dia_18/Alfredo_Casado-SCF_-_Rest_Web_Services.pdf

[33] Building Web Services the REST Way
http://www.xfront.com/REST-Web-Services.html

[34] REST Vs Web Services
http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

[35] ESB
http://icomparable.blogspot.com/2009/04/que-es-un-esb-enterprise-service-bus.html